Títol del PFC:	Llibreria d'àudio per a un instrument musical basat en Arduino
Titulació:	Grau en Enginyeria de Sistemes Audiovisuals
Autor:	Oriol Corroto Sales
Director:	Ignasi Esquerra Llucia
Data:	21 d'Octubre de 2014
Centre:	Escola d'Enginyers de Terrassa (EET)
Universitat:	Universitat Politècnica de Catalunya (UPC)

Escola d'Enginyeria de Terrassa

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Llibreria d'àudio per a un instrument musical basat en Arduino

Agraïments

En primer lloc, agrair al professor Ignasi Esquerra la possibilitat de poder dur a terme aquest projecte sota la seva tutorització, la confiança dipositada en mi i l'assessorament que ha dut a terme en tot moment que jo he necessitat.

També agrair a Óscar Martínez Carmona tot el seu ajut, els coneixements compartits, el material cedit per a la realització d'aquest projecte i l'amistat que mantenim, ja que sense ell, aquest projecte no hagués sigut capaç de dur-lo a terme.

Als meus pares, pel suport i l'ajut en tot moment durant el transcurs d'aquesta carrera i els ànims transmesos per a seguir endavant tant en els estudis com en la vida en general.

A la meva parella, Magdalena González, no només per la seva comprensió durant el temps que he dedicat a aquest projecte, sinó també per l'interès, el recolzament i el seu suport incondicional.

Índex

Llistat de figures

Introducció

Descripció del projecte

El projecte consisteix en la implementació d'una llibreria orientada a objectes amb la finalitat de ser utilitzada en la creació de nous instruments musicals utilitzant la tecnologia Arduino, i la mostra de la seva utilitat dissenyant i desenvolupant un nou instrument com a exemple.

Motivació personal

La idea de dur a terme aquest projecte sorgeix de la iniciativa de l'amic i l'ex-alumne de la Escola d'Enginyers de Terrassa Óscar Martínez Carmona, de crear instruments que es puguin tocar intuïtivament, sense la necessitat prèvia d'adquirir unes nocions musicals elevades.

A partir del seu objectiu proposat, va crear l'unoStringSynth, el primer prototip d'un baix d'una sola corda basat en la tecnologia Arduino, que més endavant es detalla en aquest informe.

Davant les possibilitats de la seva creació, i la necessitat de mostrar-lo a diverses fires, inicialment em va demanar com a músic, testejar-lo, i acompanyar-lo a aquestes presentacions per a poder tocar l'instrument en directe mentre ell exposava el seu invent. I més endavant va crear Rockin' Tech-Projects, una iniciativa de creació d'instruments musicals basats en arduino, on durant uns mesos vaig estar col·laborant, desenvolupant aquest projecte, per a fer més funcional el codi inicial de l'unoStringSynth, i adaptar-lo implementat a orientació d'objectes, per a extrapolar la funcionalitat d'aquest invent a futures idees reutilitzant el codi, i així només focalitzar en la part d'interacció humana amb l'instrument.

Com a músic i futur enginyer, la idea de crear un instrument musical nou, relacionat directament amb la tecnologia, i l'aprenentatge d'un nou llenguatge de programació com és l'Arduino, han fet que finalitzi aquest projecte, desenvolupant l'echoTheremin, un instrument basat amb sensors de proximitat per ultrasons que serveix com a un exemple de les diverses funcionalitats d'aquesta llibreria desenvolupada.

Objectius

L'objectiu principal del present projecte és dissenyar i desenvolupar una llibreria en C++ la qual anomenarem RTPlibrary[1], on s'allotgen diverses funcions implementades a mode d'eines per al desenvolupament de nous instruments musicals, i dissenyar i crear un nou instrument aprofitant aquesta llibreria, per tal de demostrar la seva utilitat.

[1] Seguint la Iniciativa Rockin' Tech-Projects

Anàlisi d'antecedents i viabilitat

Creació del unoStringSynth

Motivació

Al 2012, l'ex-alumne de la Escola d'Enginyers de Terrassa Óscar Martínez Carmona, decideix aprendre a programar de manera autodidacta Arduino, i així, dissenyar i desenvolupar un instrument capaç de ser tocat sense tenir nocions musicals i que soni seguint uns paràmetres melòdics. Un any després 2013, finalitza l'unoStringSynth, un baix elèctric híbrid d'una sola corda, capaç de reproduir tot el rang auditiu humà a través de diverses escales musicals.

fig. 1: unoStringSynth

Fonaments Teòrics

Per a aconseguir el principal objectiu de l'instrument, que és que una persona sense nocions musicals prèvies, sigui capaç d'interpretar una progressió harmònica i melòdica correcte, és necessari prèviament un coneixement aproximat de les relacions freqüencials de les notes musicals, i el funcionament de les escales musicals. Principalment, els coneixements que, en aquest cas, el intèrpret no haurà de necessitar per a poder tocar l'instrument.

Relació entre notes i freqüències

Quan s'anomena el concepte "d'octava" musical, es refereix a una proporció 2:1 a nivell freqüencial (per exemple, d'un A4 equivalent a 440Hz a un A5 equivalent a 880Hz), i tanmateix, aquesta octava es subdivideix en 12 semitons de mateixa longitud. Conseqüentment, cada semitò té una relació aproximada de $2^{1/12}$. Per convenció, la afinació europea des de 1955 acceptada per la Organització Internacional d'Estandardització (ISO) com a ISO 16, es centra en el A4 a 440Hz. A partir d'aquestes dades, es pot dur a terme el càlcul corresponent per a obtenir la correspondència freqüencial d'una escala cromàtica:

$$Freqüència = valor\ de\ referència\ d'afinació \times 2^{x/12}$$

si agafem com a eix referencial per a aquest exemple, A4, i per tant establim:

$$A4 = 0, A\#4 = 1, B4 = 2, C5 = 3, C\#5 = 4, D5 = 5, D\#5 = 6, E5 = 7, F5 = 8, F\#5 = 9,$$

$$G5 = 10, G\#5 = 11, A5 = 12$$

Podem calcular d'una manera directa la seva relació freqüencial:

A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5	G5	G#5	A5
0	1	2	3	4	5	6	7	8	9	10	11	12
440 Hz	466,16 Hz	493,88 Hz	523,25 Hz	554,36 Hz	587,33 Hz	622,25 Hz	659,25 Hz	698,46 Hz	739,99 Hz	783,99 Hz	830,61 Hz	880 Hz

A més, aquesta relació esquematitzada de L'escala cromàtica, també es pot dur a terme d'una manera senzilla i lineal en la estandardització MIDI.

El protocol MIDI no transmet senyals d'àudio, sinó dades i missatges controladors que es poden interpretar de manera arbitrària, en funció de la programació del dispositiu que ho rebi. És a dir, els valors enviats són interpretats per a la màquina que els rep, com a una partitura de valors numèrics que van del 0 fins al 127, amb informació de quan generar cada nota i les característiques que ha de tenir el so emès.

A partir de la primera especificació estandarditzada del MIDI (Musical Instrument Digital Interface) a mitjans de 1983 proposada per Dave Smith, president de la companyia Sequential Circuits dirigint-se a la Audio Engineering Society el 1981, va quedar estipulat com a eix referencial el A4 al valor 69.

Això genera la possibilitat de crear una taula desenvolupada, on es poden relacionar totes les notes musicals de 8 octaves, tant freqüencialment com pel seu valor MIDI, juntament amb el seu període ($Període = 1/freq$), la posició en un teclat i la clau a la qual pertanyen[2].

Diferents tipus d'escales

En general, una escala musical és un conjunt de sons endreçats com a notes en un entorn sonor particular, de manera simple i esquemàtica. Aquests sons estan disposats de forma ascendent, tot i que, de manera complementària, també de forma descendent, un a un en posicions específiques dins de L'escala, les quals són anomenades graus[3].

Dit d'un altre manera més col·loquial, aquests graus equivaldrien a diferents combinatòries de diversos salts que es poden dur a terme en els 12 semitons que existeixen entre octaves, dins de les escales occidentals o més usuals en el nostre entorn. Aquests salts es descriuen amb els termes Tons (T, 1 to = 2 semitons) i Semitons (st).

A partir d'aquí, l'ús d'intercalar tons amb semitons dins d'una progressió ascendent, genera diferents graus o modalitats, les quals mostrem les més habituals seguidament:

Cromàtica: st st st st st st st st st st st st

L'escala cromàtica completa representa la successió ascendent dels dotze semitons continguts dins d'una octava en un sistema atonal de temperament just.

[2] Informació adjunta a l'annex 1

[3] Graus: I tònica, II supertònica, III mediant, IV subdominant, V dominant, VI superdominant, VII sensible

Jònica (o escala major natural): T T st T T T st

L'escala jònica es la que estipula el model d'escala major. Es caracteritza per tenir un semitò a la tercera i la quarta, i entre la setena i la tònica. L'escala sense alteracions es construeix començant en la nota C.

Dòrica: T st T T T st T

L'escala dòrica és una escala menor, amb la diferència de que té una sisena major en ves de menor. Els seus semitons es situen entre el segon i el tercer grau, així com entre el sisè i el setè. L'escala sense alteracions es construeix començant en la nota D.

Frígia: st T T T st T T

L'escala frígia és una escala menor, amb la diferència de que té una segona menor en ves de major. Els seus semitons es situen entre el primer grau i el segon, i entre el cinquè i el sisè. L'escala sense alteracions es construeix començant en la nota E.

Lídia: T T T st T T st

L'escala lídia és una escala major, amb la diferència de que té una quarta augmentada en ves d'una quarta justa. Es caracteritza per disposar d'un semitò en el quart i el cinquè grau, així com entre el setè i el vuitè. L'escala sense alteracions comença en la nota F.

Mixolídia: T T st T T st T

L'escala mixolídia és una escala major amb la diferència de tenir una setena menor en ves de major. És la més coneguda de les escales gregorianes després de la major (jònica) i la menor (eòlica). És caracteritza per tenir un semitò entre la tercera i la quarta, i entre la sisena i la setena. L'escala sense alteracions es construeix començant en la nota G.

Eòlica (o escala menor natural): T st T T st T T

L'escala eòlica és la que estipula el model de escala menor. Els seus semitons es situen en el segon i el tercer grau, així com entre el cinquè i el sisè. L'escala sense alteracions es construeix començant en la nota A.

Lòcria: st T T st T T T

L'escala lòcria és una escala menor amb la diferència de tenir una segona menor en ves de major i una cinquena disminuïda en ves de la cinquena justa. Els seus semitons es situen entre la tònica i el segon grau, així com entre el quart i el cinquè. L'escala sense alteracions comença en la nota B. És considera una escala disminuïda perquè al mesurar la cinquena des de la tònica, es tracta d'una cinquena disminuïda. És L'escala més inestable de totes, ja que a més, la següent tercera després de la menor també és menor, lo que dona lloc a un acord disminuït (la sèptima és menor).

Pentatònica: T T $\frac{3}{2}$T T $\frac{3}{2}$T (major) // $\frac{3}{2}$T T T $\frac{3}{2}$T T (menor)

L'escala pentatònica és la més simple i la més utilitzada en estils com el blues, el rock i el heavy metal. Només tenen cinc notes, separades per intervals de segona major o tercera menor, sense poder haver dos intervals de tercera major junts. Existeix en modalitats majors i menors. Com a curiositat, també és anomenada com escala asiàtica, ja que és present en gran part de la música tradicional xinesa i japonesa. Per exemple, els sons fonamentals presos en els cinc orificis de la flauta tradicional japonesa anomenada Shakuhachi formen una escala pentatònica menor, i dins de L'escala "yo" utilitzada pels cants budistes Shomyo i la música Gagaku de la cort imperial del Japó és L'escala pentatònica sense semitons, corresponent al quart mode de L'escala pentatònica major.

Escala de Blues: $\frac{3}{2}$T T st st $\frac{3}{2}$T T

L'escala de Blues és la que es sol utilitzar en el rock modern. Consisteix en una escala pentatònica menor a la que s'afegeix la cinquena disminuïda o quarta augmentada com a nota de pas (blue note). És també habitual afegir altres dues notes de pas, la tercera major i la setena major.

Escala menor harmònica: T st T T st $\frac{3}{2}$T st

L'escala menor harmònica és una escala musical occidental utilitzada dins del context de la música tonal. Consisteix en una alteració del mode eòlic o escala menor natural. Per a generar un acord dominant construït a partir del cinquè grau de L'escala (necessari dins del sistema tonal), s'altera el setè grau de L'escala augmentant mig to, aconseguint així una escala menor harmònica. Té la particularitat de contar amb un interval de segona augmentada (to i mig) entre el sisè i el setè grau de l'escala.

Escales exòtiques:

Es denominen "escales exòtiques" totes les escales que no provenen directament de l'àmbit musical europeu, o que es surten de les més utilitzades del barroc fins el impressionisme, tot i que tinguessin un origen europeu. Principalment, és tracta de totes les escales que es surtin de les característiques habituals de "major", "menor" o cromàtica. Aquestes també són anomenades "escales ètniques", ja que reben noms de diferents nacionalitats, en funció principalment del seu origen. Algunes, es mostren a continuació, construïdes des de la nota C:

Àrab	C D E F Gb Ab Bb C
Bizantina	C Db E F G Ab Bb C
Hexàtona	C D Gb Ab Bb C
Espanyola	C Db E F Gb Ab B C
Hawaiana	C D Eb G A C
Hindú	C D E F G Ab Bb C
Hongaresa major	C D# E F# G A Bb C
Hongaresa menor	C D Eb F# G Ab B C
Oriental	C Db E F Gb A Bb C
Persa	C Db E F Gb Ab B C
Xina (Pentàfona)	C D E G A C

Càlcul de notes en funció d'escales

Tot aquest seguit d'escales descrites anteriorment, poden ser desglossades en semitons i salts de semitons, per lo que cada salt, i per tant, cada nota, se li pot assignar un número dins d'un array de 12 posicions (equivalents als 12 tons que poden haver-hi com a màxim dins d'una octava, segons la música occidental).

Nom escala	Distribució tonal	Valors assignats en array
Cromàtica	st st st st st st st st st st st st	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
Jònica	T T st T T T st	0, 2, 4, 5, 7, 9, 11, 12
Dòrica	T st T T T st T	0, 2, 3, 5, 7, 9, 10, 12
Frígia	st T T T st T T	0, 1, 3, 5, 7, 8, 10, 12
Lídia	T T T st T T st	0, 2, 4, 6, 7, 9, 11, 12
Mixolídia	T T st T T st T	0, 2, 4, 5, 7, 9, 10, 12
Eòlica	T st T T st T T	0, 2, 3, 5, 7, 8, 10, 12
Lòcria	st T T st T T T	0, 1, 3, 5, 6, 8, 10, 12
Pentatònica menor	$\frac{3}{2}$T T T $\frac{3}{2}$T T	0, 1, 5, 7, 8, 12
Blues	$\frac{3}{2}$T T st st $\frac{3}{2}$T T	0, 3, 5, 6, 7, 10, 12
Escala Menor Harmònica	T st T T st $\frac{3}{2}$T st	0, 2, 3, 5, 7, 8, 11, 12
Spanish Gipsy	st $\frac{3}{2}$T st T st T T	0, 1, 4, 5, 7, 8, 10, 12
Hawaian	T st T T T T st	0, 2, 3, 5, 7, 9, 11, 12

Si tornem a recordar la formula anteriorment esmentada per a obtenir el valor freqüencial d'una nota:

$$Freqüència = valor\ de\ referència\ d'afinació \times 2^{x/12}$$

i aprofitant els diversos valors de les arrays de cada escala per a substituir en el valor X, multiplicant el coeficient obtingut per una nota tonal (equivalent a la tònica o valor de referència), podem calcular automàticament el valor freqüencial de cada nota en cada escala.

Per tant, només cal desenvolupar un algoritme capaç de dur a terme aquest càlcul per a generar una senyal de sortida equivalent a la nota corresponent, a partir de dues entrades, una que senyali el valor de la tònica, i l'altre equivalent al punter dins de la array seleccionada.

Disseny i desenvolupament tècnic

A partir de la teoria plantejada en l'apartat anterior, l'Óscar Martínez va decidir implementar un instrument que treballés sota aquest patró de càlcul automatitzat en una corda de baix elèctric, ja que era el tipus d'instrument que ell estava més familiaritzat.

fig. 2: màstil unoStringSynth

Va aconseguir un baix elèctric de segona mà i va manipular el màstil i la pala per a fer-lo fretless i que només disposés d'una sola corda centrada, situada a sobre d'una resistència tàctil, que varia la tensió de sortida d'un circuït simple connectat a una placa Arduino Uno. L'objectiu d'aquesta resistència tàctil (anomenada SoftPot de la marca Sparkfun) és senyalar la nota de cada array corresponent a L'escala musical seleccionada, per a que la placa Arduino generi una ona quadrada amb la freqüència corresponent, a més de la seva ordre MIDI.

Per altra banda, amb la finalitat de poder manipular l'inici, la intensitat i el final de cada nota de la mateixa manera que es duu a terme amb un baix elèctric, va col·locar la pastilla inductiva de l'instrument paral·lelament a la corda, i va dissenyar un senzill circuit demodulador d'envolvent, per a poder enviar la senyal de sortida, equivalent a la intensitat recollida per la corda pinçada, a la placa Arduino Uno. Així, aquesta pot recollir la quantitat de tensió mapejada en funció del valor màxim i mínim que pugui recórrer la pastilla (valors prèviament calibrats a cada encesa de l'aparell), i després transformar-ho en valors d'intensitat de sortida de la senyal quadrada generada per la pròpia placa.

fig. 3: placa Arduino Uno interna de l'unoStringSyntg

Finalment, la selecció de les escales generades i la freqüència establerta com a tònica o fonamental, es duia a terme mitjançant dos potenciòmetres rotatius de 12 posicions, on un escollia L'escala, i l'altre la tonalitat. També va implementar un circuit format per dos polsadors i 4 leds indicadors, que oferien la opció de seleccionar la quantitat d'octaves en la que es dividia el recorregut de la resistència tàctil.

Codi

A partir d'aquest punt, és quan s'inicia el projecte que mostra aquest informe. Un cop l'Óscar Martínez va dissenyar i implementar el codi arduino de l'unoStringSynth[4] , el seu inventor i servidor, vam re-analitzar-lo, comentant pas a pas el procés de tots els components i línies de codi, i mirant la manera de fer-lo més eficient o bé menys redundant[5].

El principal objectiu d'aquest anàlisis dut a terme, va ser la familiarització per part meva amb la programació en Arduino, i la comprensió exacte del funcionament de l'unoStringSynth, per tal de poder fer un primer plantejament de possibilitats per a orientar el codi a objectes.

Algunes de les millores durant aquest primer anàlisis van ser:

1- Comentarització de tot el codi, amb la finalitat de fer-lo més entenedor i agradable davant la seva possible futura manipulació.
2- Implementació de L'escala cromàtica amb els seus coeficients harckodejats
3- Eliminació de línies de codi que no s'utilitzaven
4- Millorar la calibració dels rangs de treball dels diferents sensors mitjançant la funció "map" del llenguatge Arduino

Naixement de Rockin' Tech-Projects

A partir d'aquest projecte, Òscar Martínez decideix crear la iniciativa Rockin' Tech-Projects, a mode de laboratori creatiu dins del col·lectiu maker de Barcelona, amb la finalitat de poder intercanviar coneixements amb altres persones introduïdes en la tecnologia Arduino i en el moviment "do it yourself"[6], desenvolupant nous projectes i millorant els prototips ja creats.

fig. 4: logotip Rockin' Tech-Projects

Per aquest motiu, a mode de col·laboració dins d'aquesta iniciativa, es va decidir dirigir aquest projecte a la creació d'una llibreria per a arduino orientada a objectes, per a millorar i facilitar la creació i desenvolupament dels projectes de Rockin Tech-Projects. I per això, la llibreria rebrà el nom de RTPlibrary.

[4] Codi Arduino recollit a l'annex 2
[5] Informació ampliada a l'annex 3
[6] Cultura contemporànea relacionada amb la enginyeria i la tecnologia, on l'individu crea les seves propies eines a partir de la necessitat de crear i innovar de manera creativa, a través de la invenció i creació de prototips.

Implementació de la Llibreria RTP

Objectiu de la implementació (orientació a objectes)

Un cop analitzat i estructurat el codi font de l'unoStringSynth, la següent tasca duta a terme va ser la descomposició dels diferents elements i funcions que hi treballen, a fi de clarificar les possibles eines que caldrien per a refer l'unoStringSynth a través d'una llibreria. Inicialment, es procedeix a la següent estructuració de necessitats:

- Càlcul de notes dins de la Matriu de coeficients
- Càlcul de la freqüència de la nota tònica
- Captura de la interacció humana física que estableix quina nota es vol dur a terme
- Captura de la interacció humana física que estableix a quina intensitat es vol dur a terme
- Captura de la interacció humana física que estableix sota quina escala es vol treballar
- Calibratge dels sensors
- Enviament de missatges MIDI corresponents a les notes emeses

Llibreria RTPlibrary

RTPDiatonicMatrix

L'eina RTPDiatonicMatrix està formada pels fitxers RTPDiatonicMatrix.h i RTPDiatonicMatrix.cpp[7]. És la principal responsable de decidir quin serà el coeficient que multiplicarà a la tònica per a calcular la freqüència de la nota de sortida.

Està formada principalment per una matriu de N escales X M coeficients, on M varia en funció dels salts que disposa l'escala, i un seguit de get's i set's que modifiquen els paràmetres necessaris per a el retorn de la nota seleccionada. Dins d'aquesta matriu, s'han substituït els coeficients hard-codejats [8]per als valors de l'1 al 12, per tal de poder dur a terme el càlcul a temps real de la freqüència i així evitar arrossegar un possible error decimal en les operacions.

També s'ha afegit el vector "scaleName", per tal de que estigui relacionada cada escala amb el seu nom, en cas de voler ser imprès en pantalla en el moment de ser seleccionada.

[7] Informació ampliada a l'annex 4 i 5

[8] Dades incrustades directament dins del codi del programa, sense obtenir-los a partir d'un càlcul

RTPSelectTone

L'eina RTPSelectTone està formada pels fitxers RTPSelectTone.h i RTPSelectorne.cpp[9]. És la encarregada de decidir la tonalitat de L'escala que es vulgui executar.

Està formada per un vector amb els salts de semitò d'una octava que van del 0 a l'11, i els seus respectius get's i set's per a obtenir les dades d'input per part de l'usuari quan decideix la tonalitat, per així obtenir el coeficient a multiplicar amb la freqüència d'afinació i aconseguir la freqüència exacta de la tònica amb la que es vol treballar.

També està acompanyada del vector "toneName", que atorga el nom de les notes a cada freqüència corresponent, en el cas de que es vulgui imprimir en pantalla.

RTPPitchControl

L'eina RTPPitchControl està formada pels fitxers RTPPitchControl.h i RTPPitchControl.cpp[10]. La seva funcionalitat és, com bé indica el seu nom, efectuar un canvi de pitch a la nota que s'estigui executant.

La necessitat d'implementar aquesta eina esdevé en el moment en que al projecte unoStringSynth, se li amplia la funcionalitat de dur a terme aquest efecte utilitzant un sensor de doble eix i una palanca, a mode de Floyd Rose[11].

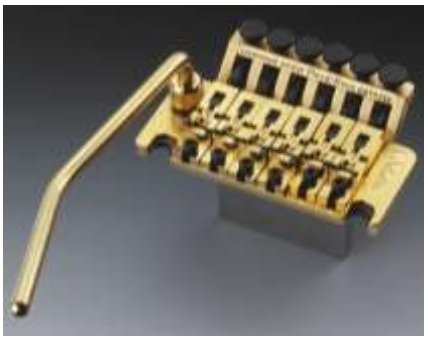


fig. 5: Floyd Rose

RTPMidiSet

L'eina RTPMidiSet està formada pels fitxers RTPMidiSet.h i RTPMidiSet.cpp[12]. Consisteix en un conjunt de get's i set's que obtenen els valors de nota, intensitat i velocitat, a més del note-on/note-off, per a traduir aquests valors al missatge midi corresponent.

Aquesta eina utilitza dins seu la eina RTPMaths, per a poder dur a terme el càlcul de logaritmes neperians, ja que en C++ aquesta funció no està carregada per defecte.

[9] Informació ampliada a l'annex 6 i 7
[10] Informació ampliada a l'annex 8 i 9
[11] Pont de guitarra elèctrica flotant que permet el canvi de la afinació de l'instrument mitjançant una palanca.
[12] Informació ampliada a l'annex 10 i 11

RTPMaths

L'eina RTPMaths està formada pels fitxers RTPMaths.h i RTPMaths.cpp[13]. Dins seu només allotja una funció que retorna càlculs de logaritmes neperians, però la idea de la funcionalitat d'aquesta eina, és poder implementar altres càlculs que puguin ser útils per a la conversió de valors en qualsevol moment del procés de l'algoritme, en funció de les necessitats de cada instrument.

Ús de la llibreria per a la creació d'un nou instrument: echoTheremin

A mida que es va anar desenvolupant la llibreria, es va arribar a la conclusió, que la millor manera de demostrar la seva utilitat, i el motiu pel qual dur a terme aquest projecte, era desenvolupar un nou instrument, que seguís les mateixes directrius que l'unoStringSynth, però que a nivell d'interacció física no s'assemblés.

Per aquest motiu, després de plantejar diferents idees i alternatives, es va decidir decantar-se pel disseny d'un instrument amb el que es pogués dur a terme interpretacions musicals d'una manera semblant a la tècnica que es fa servir amb els Theremin, però d'una manera molt més intuïtiva i senzilla si es compara amb la complexa tècnica d'execució dels Theremin.

Aprofitant la mateixa filosofia, l'instrument ha de ser capaç de crear melodies dins d'una harmonia de manera prou intuïtiva, com per a que una persona que no disposi de nocions musicals, pugui dur a terme execucions de dificultat mitja-alta amb un instrument convencional.

Per això, es decideix aprofitar la mateixa tecnologia Arduino, establint la interacció física de l'intèrpret amb sensors de posició per ultrasons. D'aquí, el seu nom: echoTheremin.

En aquest cas, les eines que fa servir l'echoTheremin són la RTPDiatonicMatrix i la RTPSelectTone, deixant oberta la ampliació de prestacions de l'instrument utilitzant les altres eines implementades, introduint-les dins del codi .ino del echoTheremin[14].

[13] Informació ampliada a l'annex 12 i 13
[14] Informació ampliada a l'annex 14

Desenvolupament Tècnic echoTheremin

El referent: el Theremin

El Theremin és un instrument musical creat pel físic rus Lev Termen al 1919. Consisteix en dos antenes orientades perpendicularment entre elles, on es reparteixen les tasques de selecció de nota i intensitat d'aquesta.

Cadascuna de les antenes genera ones electromagnètiques d'intensitat molt dèbil, de manera que al apropar una mà, que fa de conductor elèctric, altera el camp electromagnètic que genera l'antena canviant la seva capacitància, i per tant, modificant la freqüència de la corrent alterna que genera l'instrument. Aquesta senyal és demodulada per a poder obtenir sons en un rang freqüencial audible[15].

El mètode d'execució de l'echoTheremin, vol imitar a aquest instrument casi centenari, substituint les antenes electromagnètiques per sensors de proximitat per ultra-freqüència i tota la circuiteria analògica que demodula la senyal, per una placa arduino.

fig. 6: Lev Termen utilitzant un Theremin

Arduino

Arduino és una eina compatible per a qualsevol tipus de computadora capaç de detectar i processar variacions físiques externes. Principalment és tracta d'una plataforma de computació física de codi obert basada en una placa electrònica senzilla i un entorn de desenvolupament per a escriure-hi software destinat a aquesta.

fig. 7: logotip Arduino

Les seves utilitats són diverses, però es centren en el desenvolupament d'objectes interactius, utilitzant a les entrades qualsevol tipus d'interruptor o sensor, i les sortides per a controlar llums, motors, o com és aquest cas, fins i tot sons.

A més, el codi és obert i es pot descarregar de manera gratuïta, fet que fa que pugui ser millorat i adaptat a comunicar-se amb altres softwares que s'estiguin executant a l'ordinador (com per exemple: Flash, Processing, MaxMSP, Matlab...), i les plaques es poden muntar a mà o bé comprar-les prefabricades.

[15] de 20Hz a 20KHz

Historia

Aquesta tecnologia es va crear al 2005 com a projecte per als estudiants a l'institut IVREA (Itàlia), per tal d'oferir una alternativa als microcontroladors de cost massa elevat (uns 100$) que aleshores utilitzaven. El professor de l'institut IVREA Massimo Banzi, juntament amb la col·laboració de l'estudiant colombià Hernando Barragán, van desenvolupar un llenguatge de programació i una plataforma de desenvolupament, que no fos més costosa de 30€, plug&play i que fos compatible amb totes les plataformes informàtiques com OSX, Windows o GNU/Linux.

Actualment aquesta tecnologia de prototipatge està sent un èxit en ventes, principalment per la seva simplicitat per a ser programada en codi obert, i pel reduït cost que suposa. Comercialitzen des de la seva pàgina web i subministren els diferents models que han evolucionat en funció de les necessitats de cada projecte a botigues de components electrònics de tot el món.

Models i prestacions

A partir del seu èxit, Arduino ha anat perfeccionant i creant nous models de plaques, adaptant cada model a les possibles necessitats que puguin haver a cada projecte. Des d'un o diversos processadors, transmissió bluetooth, fins a afegir-hi rodes a la pròpia placa per a casos de robòtica o dissenys de mides molt reduïdes per a economitzar espai.

Per a la elaboració de l'echoTheremin, s'ha decidit decantar-se pel model Leonardo, que ofereix les següents prestacions:

- Microcontrolador ATmega32u4
- 20 pins digitals I/O[16]
    - 7 poden ser utilitzades per a sortides PWM[17]
    - 12 poden ser utilitzades com a entrades analògiques
    - SDA i SCL, per a la comunicació amb perifèrics I2C[18]
- Oscil·lador de 16MHz
- Connexió micro-usb
- Connector d'alimentació AC/DC
- Botó de reseteig

[16] Inputs/Outputs (Entrades/Sortides)
[17] Pulse-width modulation
[18] Inter-Integrated Circuit. Tipus de bus de comunicacions en serie basat en una via per a transmetre la informació i l'altre per a la senyal del rellotge.

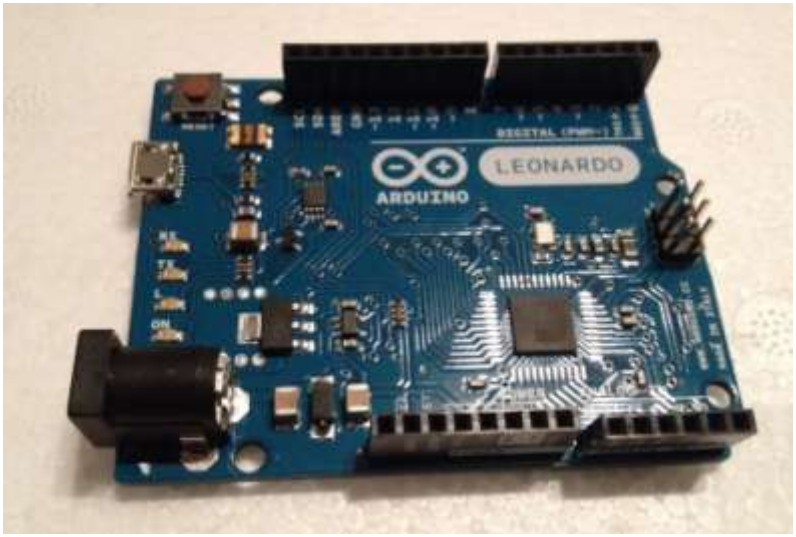


fig. 8: placa Arduino Leonardo escollida per al desenvolupament de l'echoTheremin

El principal motiu pel qual s'ha decantat per a aquest model, i no per al model Uno (que utilitza l'unoStringSynth), és per la major versatilitat d'entrades i sortides, oferint la possibilitat d'ampliar les prestacions del propi instrument, i per la comunicació I2C, ja que la placa ofereix d'una manera molt còmode la instal·lació d'aquest tipus de perifèrics, com poden ser mòduls de display, per a obtenir informació de paràmetres.

Altres components

Polsadors 05A/250V

Al analitzar detalladament el projecte unoStringSynth, un dels inconvenients que se li trobava, era la limitació física dels potenciòmetres rotatius que seleccionaven tant l'escala com la tonalitat.

Al estar fixats en un nombre limitat d'eixos, en el cas de voler actualitzar el software per a canviar o incrementar les prestacions, existia un nombre limitat d'escales i tonalitats en el moment d'execució per part de l'interpret.

Utilitzant polsadors com a booleans per a poder moure's dins de les matrius d'aquests paràmetres, l'única limitació és la capacitat d'emmagatzematge de la pròpia placa Arduino.

Per contra, prenent aquesta decisió, es perdia el control visual de la variable que s'estava escollint en cada moment. Però, utilitzant un mòdul de Display I2C , l'inconvenient queda resolt.

fig. 9: polsador

Mòdul de Display Arduino I2C 16x2 Serial Blue LCD

Per a poder visualitzar en tot moment els paràmetres que s'estan modificant durant la interpretació, es decideix instal·lar un Display I2C, que venen amb un mòdul ensamblat que facilita el connexionat i la comunicació amb la placa Arduino.

fig. 10: Display Arduino I2C 16x2 Serial Blue LCD

La principal característica del protocol de comunicació i2C és que utilitza dues línies per a transmetre la informació, una per a les dades (SDA) i l'altre per a la senyal de rellotge (SCL), per a poder sincronitzar-se amb la placa arduino. En aquest cas, és un dels principals motius de l'elecció del model Leonardo, ja que ja incorpora dues connexions específiques per a aquest tipus de protocol.

En el prototip desenvolupat, aquest display mostra en tot moment quina escala o quina tonalitat s'està escollint per a interpretar.

Base jack femella 6,3mm


fig. 11: connector jack femella

Inicialment, els primers muntatges i proves es van dur a terme utilitzant un petit altaveu rudimentari. Però, per a poder disposar d'una sortida d'àudio professional, amb la possibilitat de poder manipular el so reproduït per l'instrument amb altres aparells, cal extreure'n una senyal de línia, per tant es decanta per una connexió standard de Jack de 6,3mm, per tal de fer l'aparell compatible amb qualsevol altre component d'àudio professional amb sortida de línia.

Base femella MIDI

Una manera d'augmentar el ventall de possibilitats de qualsevol instrument digital, és que, al igual que genera sortides de tensió transformades en tons, pugui enviar senyals midi a mode de controlador, corresponents a les mateixes notes que pretén emetre. Per a aquest motiu, tot i que en el codi .ino del prototip no


fig. 12: connector MIDI femella

s'ha implementat, es decideix dur a terme la instal·lació de la sortida MIDI per així poder acabar de desenvolupar més endavant la funció per a emetre senyals MIDI, i, a més de poder utilitzar-se com a instrument musical, sigui capaç de comunicar-se amb altres aparells MIDI a mode de controlador.

Sensors de posició per ultrafreqüències

fig. 13: frontal sensor HC-SR04

Com l'objectiu és aconseguir un mètode d'execució de l'instrument similar al Theremin (és a dir, la interacció amb aquest apropant i allunyant les mans sense arribar a establir un contacte físic amb ell), ha sigut necessari instal·lar sensors de posició que substitueixin els camps electromagnètics de les antenes de to i posició, amb la finalitat d'abaratir costos i simplificar el projecte.

Per aquest motiu, i també per donar coherència al nom de l'instrument, es va decidir decantar-se per sensors de posició per ultrasons, que consisteixen en un altaveu emissor direccional i un micròfon que recull el rebot de la senyal emesa, i calcula el retard per a obtenir la distancia que el so ha recorregut enviant la diferència de temps a la placa arduino.

Entre els diversos models del mercat que ara mateix estan disponibles per internet, s'ha decidit decantar-se pel model HC-SR04 (d'un cost aproximat de 2,85€) principalment per la resolució que ofereix i pel baix cost que suposa respecte altres marques de prestacions similars com podria ser el model PING))) de la marca Parallax (d'un cost aproximat de 29,99$ a la seva pàgina web)

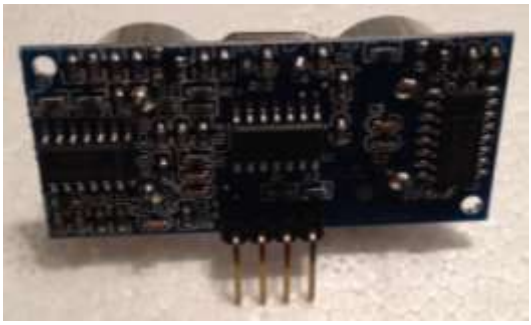


fig. 14: posterior sensor HC-SR04

Aquest model (al igual que la majoria), funciona enviant un pols de 10uS a la entrada de trigger per a que el mòdul transmissor (altaveu) enviï un feix de 8 cicles d'ultrasons a 40HHz, que després serà rebut a través del mòdul receptor (micròfon). Un cop realitzat aquest cicle, el sensor envia senyal a través de la pestanya "echo", per a poder ser calculada la distància de la posició de l'objecte que es troba enfront del sensor, mitjançant el següent càlcul:

$$\frac{uS_r - uS_t}{rang\ màxim\ del\ sensor\ (58cm)} \times \frac{velocitat\ del\ so\ (340\,{}^{m}/_{s})}{2}$$

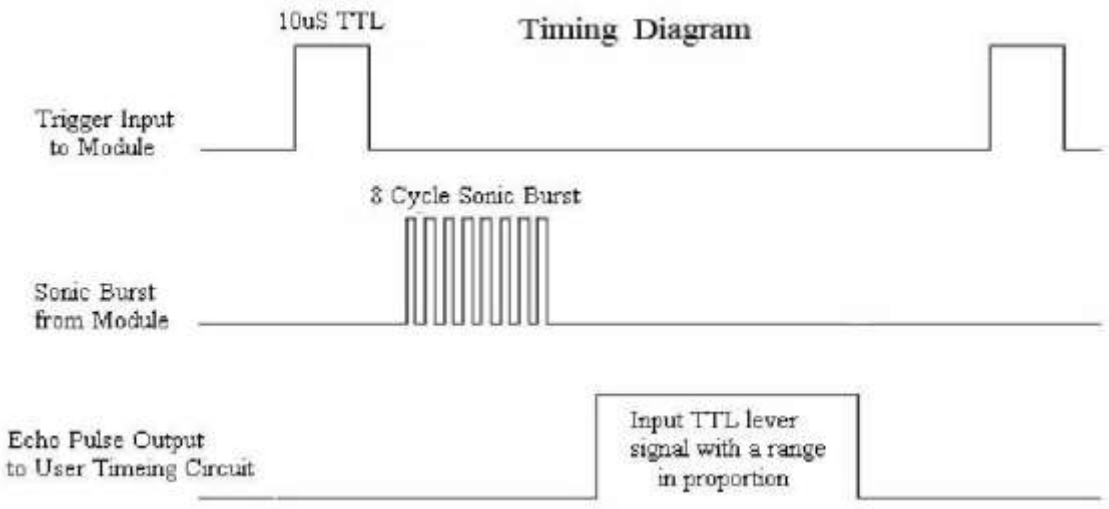


fig. 15: gràfica d'un cicle de treball dels sensor HC-SR04

Recipient

Per finalitzar el muntatge de l'instrument, és necessari d'una carcassa per a poder instal·lar tots els components, i així aconseguir que aquest sigui pràctic i senzill de fer servir, alhora que còmode de transportar i manejable per a poder utilitzar-lo sense impediments ni problemes.

fig. 16: models de recipient de porexpan

Per al prototip, s'aconsegueix una capsa de porexpan de 11cm x 17,5cm x 4,5 cm, amb espai suficient com per allotjar tots els components, escollida en aquest cas, per abaratir costos, a més de les seves avantatges davant carcasses de metall com poden ser el pes o la facilitat de manipulació.

Tot i així, si el projecte evolucionés amb la intenció de crear una producció en sèrie, seria més adient instaurar-los en carcasses metàl·liques els models finals, principalment per la seva robustesa davant cops.

Enfocat a un plantejament d'us en directe, es decideix manipular la capsa per a poder ser instal·lada en un suport de plat de bateria, oferint un ventall ampli d'opcions i comoditats d'instal·lació i execució, ja que l'intèrpret per norma general utilitzarà el seu instrument dret i no assegut, i poder disposar d'aquest fix en un sol eix, evitarà les possibles incomoditats que poden sorgir al executar melodies amb l'instrument damunt d'una taula on no hi ha suficient espai per a la ma o poden haver-hi altres objectes que dificultin i/o interfereixin als sensors de

proximitat. Per no parlar de la comoditat de transportar un suport de plat de bateria, en contra d'una taula.

Pressupost

Nº	Component	Valor/unitat	Preu total
1	Placa Arduino Leonardo	18€	18€
2	Polsador mini negre 0,5A/250V	1,16€	2,32€
1	Polsador mini vermell 0,5A/250V	1,16€	1,16€
1	Base jack 6,3mm	0,55€	0,55€
1	Base femella MIDI	1,05€	1,05€
1	Arduino I2C 16x2 Serial Blue LCD Module Dislpay Screen	5,11€	5,11€
2	Sensor de posició per ultrafreqüències HC-SR4	2,85€	5,7€
27	Jumpers	0,25€	6,75€
5	Resistències 10K	0,02€	0,1€
1	Recipient	3,5€	3,5€
		TOTAL	44,24€

Disseny i muntatge

Implementació del codi echoTheremin.ino

El codi implementat dins del fitxer echoTheremin.ino, segueix el següent diagrama de flux:

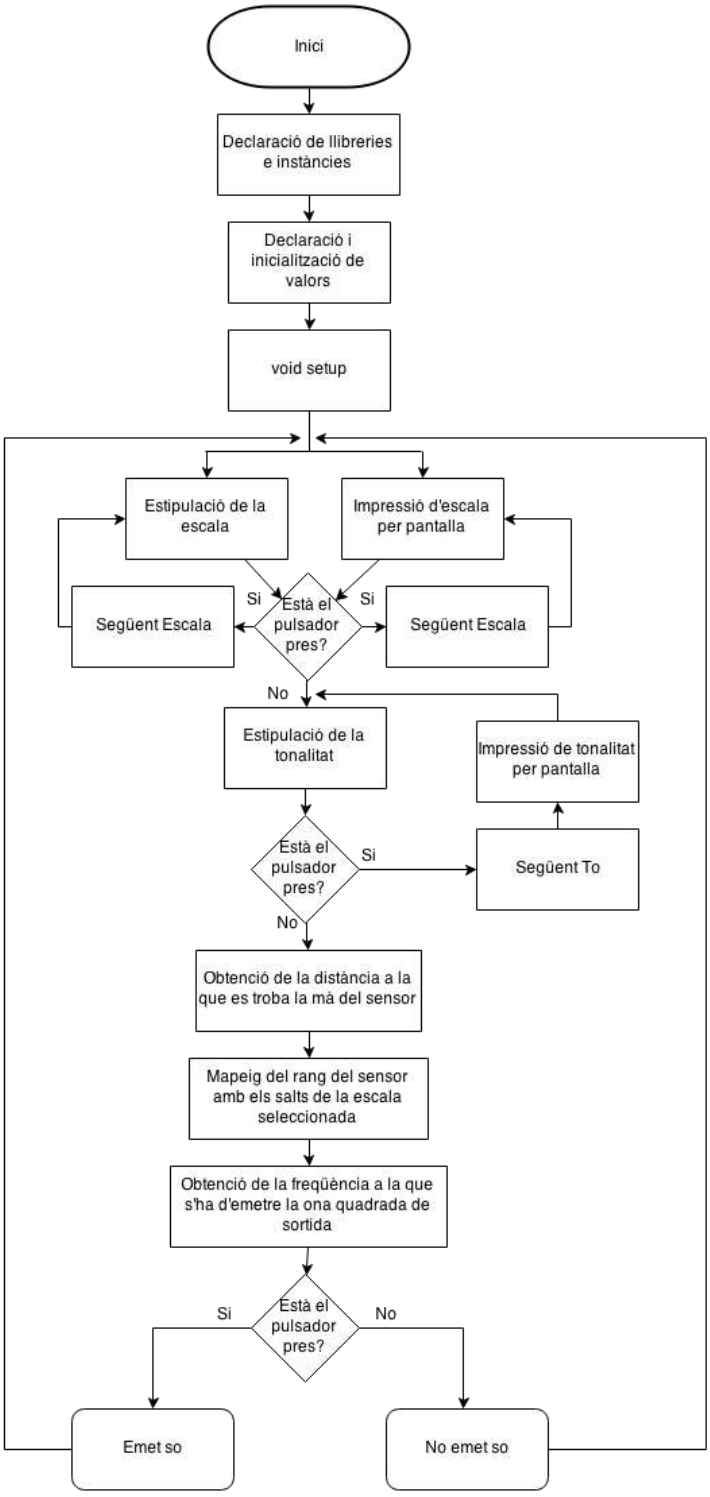


fig. 17: diagrama de flux del codi echoTheremin.ino

Un problema greu que va sorgir durant el procés, va ser el descobriment d'un error arrossegat a nivell d'incompatibilitat amb el sistema arduino i Windows 7 64-bits. Al introduir un sketch del display dins del fitxer .ino, el port sèrie de l'ordinador va deixar de ser detectat, i la placa arduino va deixar de funcionar.

Després de dies de cerca mitjançant foros especialitzats en arduino, es va trobar el mètode de resolució. Era necessari bolcar l'sketch més senzill (Blink) que ve per defecte amb arduino, i just enmig del procés, resetejar la placa utilitzant el botó de reseteig. És molt important, i en cap de les respostes va ser trobat aquest detall, tenir tots els components desconnectats just quan es du a terme aquest procés. Si es deixa algun component connectat, el port sèrie torna a desaparèixer al cap d'uns segons.

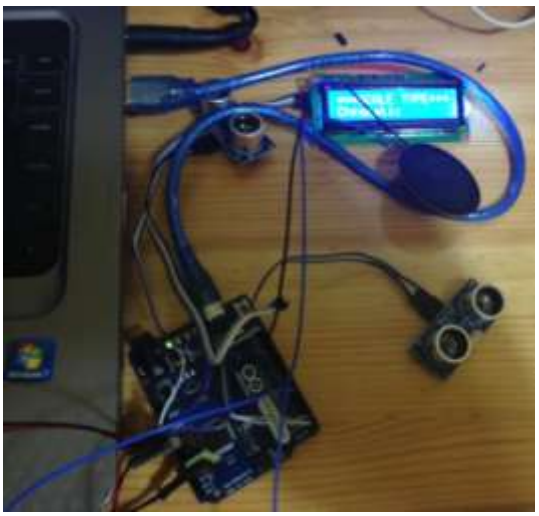


fig. 18: proves dutes a terme durant la implementació del codi a la placa Leonardo

Un altre problema ha sigut la velocitat a la qual treballen els sensors de proximitat, ja que envien masses dades per a què la placa arduino li doni temps a calcular totes les mesures reals, i sorgeixen errors de càlcul, generant falsos tons no corresponents amb el que es vol interpretar.

Una possible solució que es va intentar implementar va ser crear un buffer de 15 mesures, per tal de crear un promig i així obtenir el valor més pròxim al realment desitjat. Malauradament, aquest procés crea una latència massa elevada com per a poder dur a terme una interpretació fluida, per tant es va intentar reduir la mida del buffer a 5. Per desgràcia, tampoc s'acaben d'obtenir resultats òptims amb aquest procés, així finalment es va decantar per a afegir un delay de 25 milisegons, per tal que la placa pugui treballar amb prou marge com per què no erri en el càlcul de distància dels sensors de proximitat.

Disseny i distribució dels elements

Un cop finalitzada la implementació del fitxer .ino, i per tant, decidit i estipulat el mètode d'execució de l'ecoTheremin, s'ha procedit al disseny previ de la carcassa que allotjarà tots els components que conformen l'instrument.

S'han estudiat diverses combinacions de distribució, tenint en compte els següents factors:

- Compatibilitat entre components (no es superposin a nivell espaial entre ells, ni tampoc puguin interferir amb el funcionament dels altres)
- Practicitat: l'instrument ha de ser de fàcil transport i manipulació, tant per al moment d'utilitzar-se, com quan es vulgui manipular per a dur qualsevol reparació i/o actualització de codi.

- Intuïtiu: l'echoTheremin, al igual que el Theremin, ha de poder tocar-se amb les mans d'una manera intuïtiva, sense posicions de mans complexes o incòmodes per a qui l'hagi de fer servir.

Per aquests motius, finalment s'acaba decidint la següent configuració descrita, on:

A: Placa Arduino Leonardo
B: Sensor de proximitat per ultrasons
C: Display
D: Connector Jack
E: Connector MIDI
F: Polsador
G: Cavitat adaptada per a peu de plat de bateria
H: Connector de corrent

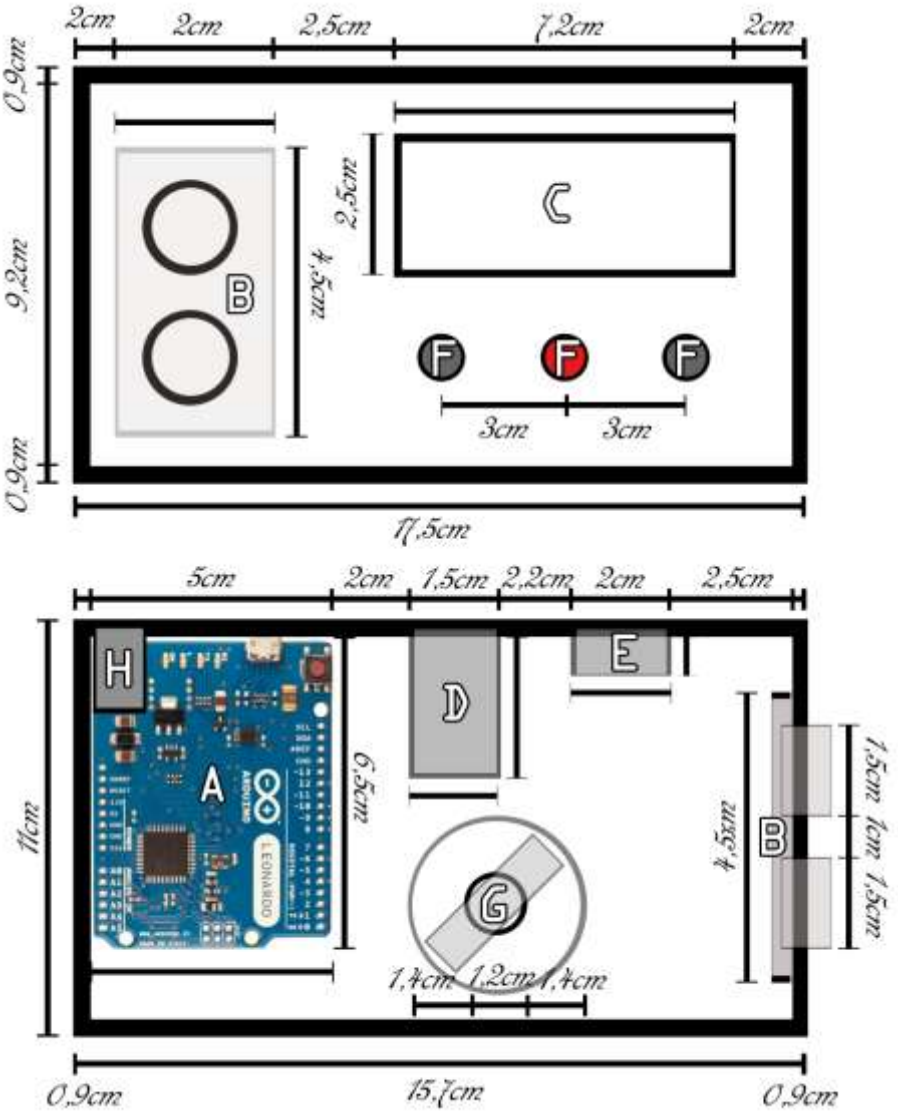


fig. 19: disseny echoTheremin

Aquesta configuració permet la fàcil manipulació i extracció, en el cas de que sigui necessari, de qualsevol dels components, només extraient la tapa.

Els dos sensors d'ultrasons, estan ubicats de manera perpendicular, per tal d'evitar que puguin interferir entre ells en la recepció de les senyals

A més, amb l'espai que no ocupen els elements electrònics, s'ha dut a terme un orifici per tal de que el propi instrument pugui ser subjectat amb un peu de plat de bateria, fent-lo així més còmode a l'hora d'haver de muntar l'instrument en un directe, i evitant els possibles problemes que pot suposar intentar executar-lo sobre una taula plana (limitació de mobilitat per part de l'intèrpret, possibles objectes que puguin interferir la recepció de senyal del sensor horitzontal...).

Muntatge

Primerament, es va procedir a dur a terme les cavitats pertinents a la capsa de porexpan, per tal d'assegurar la seva fixació i distribució, comprovant que cap element destorbés en concepte d'espai i quedessin correctament encaixats.

fig. 20: procés de muntatge de l'echoTheremin

Després es va pintar de color negre amb pintura plàstica, per tal de dissimular les imperfeccions del porexpan, i així aportar un toc més seriós i professional al prototip, tot i ser de baix cost.

Acte seguit, es van col·locar tots els elements en el seu lloc corresponent seguint amb detall el pla de disseny dut a terme. La pintura va fer que algunes cavitats quedessin reduïdes, procedint a repetir el procés en algunes cavitats concretes. Un cop corregides i pintades de nou, es van acabar d'ubicar la resta de components, i es van collar a la capsa.

Finalment, es va procedir a dur a terme el connexionat, seguint el següent esquema:

- **Vermell:** La major part dels elements electrònics, calen ser alimentats, per tant, són connectats a la sortida de tensió de 5V de la placa arduino.
- **Blau:** alguns dels components electrònics tenen com a finalitat, enviar senyals que permetin la modificació de paràmetres de configuració de l'instrument. És per això, que necessiten ser connectats a una entrada digital, on al detectar una variació de tensió, puguin executar el canvi pertinent a la variable a la qual estan destinats cadascun.
- **Negre:** tots els components electrònics necessiten ser connectats a una toma de massa per a que el circuït quedi tancat correctament, anomenada "ground". Aquestes connexions estan repartides entre les dos entrades "GND" de la placa Arduino.

- Gris: el model de sensor de proximitat HC-SR04 necessita rebre un impuls de 10uS per tal d'iniciar la emissió de la senyal de 40KHz. La placa arduino s'encarrega d'enviar aquest impuls a través de la pestanya "trigger".
- **Morat:** un cop que el sensor HC-SR04 inicia la emissió de 8 cicles d'ultrasons i rep el seu eco, la pestanya "echo" envia el temps en cicles que ha trigat des de que ha rebut l'impuls pel trigger des de la placa arduino, fins que ha rebut els 8 cicles pel mòdul receptor, per tal de que la placa arduino pugui dur a terme els càlculs pertinents i així obtingui la distància de l'objecte que s'interposa davant del sensor.
- **Verd:** és el connexionat corresponent a la connexió SDA de la comunicació i2C que segueix el display instal·lat. La seva funció és de sincronització de senyal, entre el display i l'arduino, a mode de rellotge.
- Groc: aquesta connexió correspon a la línea SDA del protocol i2C, i la seva funció és transmetre les dades que la placa Arduino vol plasmar a la pantalla.

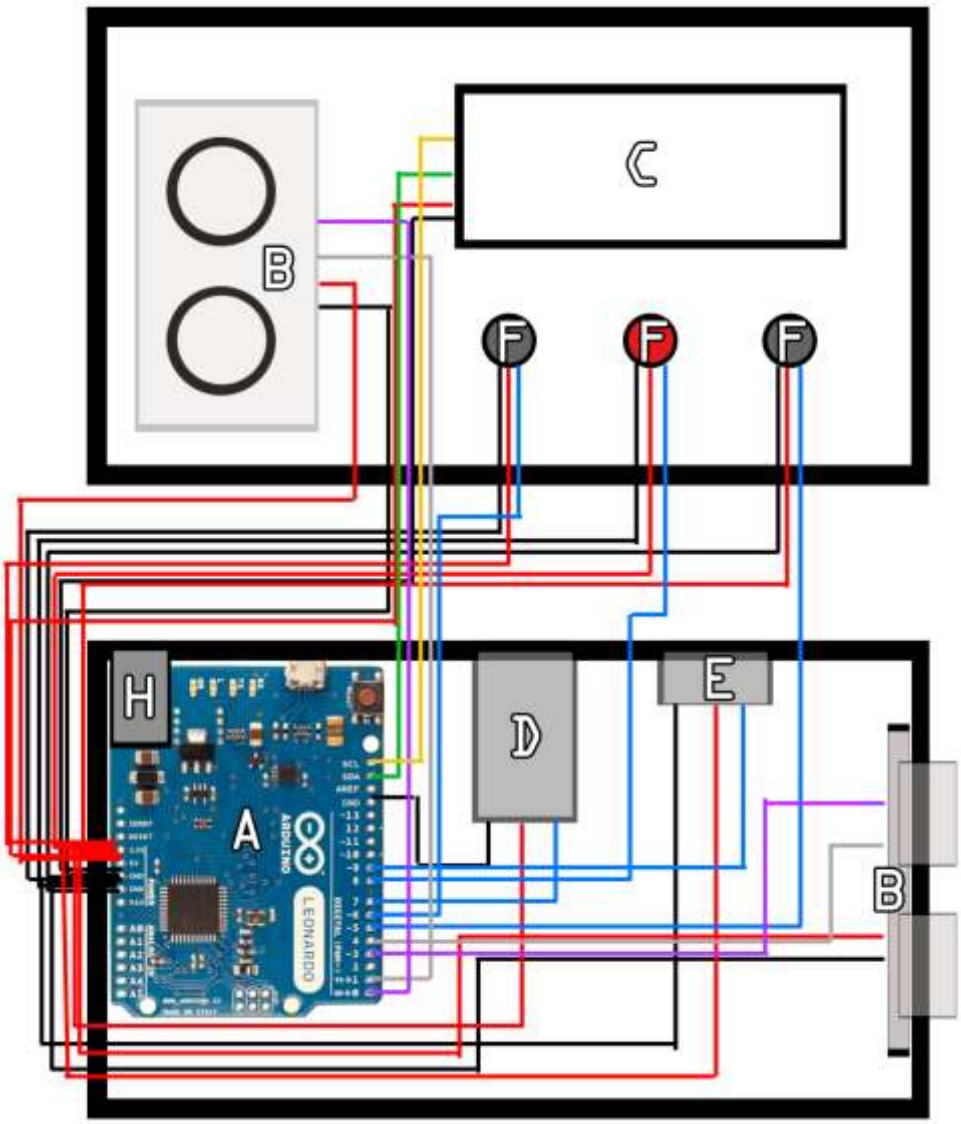


fig. 21: plànol de connexionat de jumpers dins de l'echoTheremin

Per acabar, al comprovar que tots els jumpers connectats provocaven certa pressió a la capsa, que evitava el seu tancament, es va procedir a instal·lar unes tires de velcro per tal d'assegurar el seu sistema d'obertura i tancament sense incidències.

Concordança de resultats i objectius a partir del pla de treball

Plans de treball prèviament estipulats

Pla de treball 1 (03/02/2014 – 01/06/2014)

1. **Estudi de l'estat actual. (03/02/2014-03/03/2014)**
   - 1.1 Interpretació del codi actual del uSS
   - 1.2 Investigació de la llibreria Mozzy
   - 1.3 Recerca d'altres possibles llibreries ja implementades que puguin ser útils

2. **Definició d'objectius. (04/03/2014-20/03/2014)**
   - 2.1 Planificació e brainstorming d'idees
   - 2.2 Plantejar i desenvolupar noves funcions dins del codi:
     - 2.2.1 Implementar noves escales (Arpegiador Major/menor, cromàtica...)
     - 2.2.2 Implementar comunicació MIDI
     - 2.2.3 Implementar doble funcionalitat dins de la mateixa barra resistiva tàctil

3. **Desenvolupament. (21/03/2014-21/04/2014)**
   - 3.1 Introducció a Arduino, Workshop de la mà de l'Òscar Martínez *(22/01/2014)*
   - 3.2 Separar les diferents funcions del codi del uSS per a esquematitzar la llibreria i plantejar possibles millores d'eficiència
   - 3.3 Implementar la llibreria
   - 3.4 Creació i manteniment de Github per a distribució i desenvolupament de la llibreria implementada (evolució OpenSource del codi)

4. **Proves (22/04/2014-02/05/2014)**
   - 4.1 Carrega de la nova llibreria al uSS per comprovar les millores d'eficiència i noves opcions (dur a terme comparativa entre "abans i després")
   - 4.2 Linkar Arduino amb Matlab per a obtenir simulacions i comparatives d'eficiència més precises.

5. **Memòria (03/05/2014-01/06/2014)**
   - 5.1 Dur a terme la Wiki de la llibreria implementada
   - 5.2 Redacció de la memòria
   - 5.3 Presentació del prototip uSS amb les millores implementades

Objectius assolits i conclusions

Malauradament, no es van assolir els objectius de cada etapa, i el projecte no es va finalitzar en el temps planificat. L'aprenentatge d'arduino va ser més costós del previst, i degut a una

oportunitat laboral impossible de revocar, a més de la incidència de incompatibilitat anteriorment esmentada entre Arduino i Windows 7 64-bits, es va superar el deadline estancant-se en els punts 3 i 4.

Durant aquest procés, es va decidir substituir les tasques de creació del GitHub, entrada de wikipèdia i la focalització al projecte unoStringSynth, per la aplicació directa de la RTPlibrary a la creació d'un nou instrument.

Pla de treball 2 (15/09/2014 – 20/10/2014)

1. **Recopilació de la feina feta anteriorment** (**15/09/2014-18/09/2014**)
    a. Llibreria implementada fins aleshores
    b. Part de la memòria redactada
    c. Components reutilitzables

2. **Compra dels components a reemplaçar** (18/09/2014)
    a. Llistat del material necessari
    b. Cerca d'establiments que disposin del material
    c. Elaboració d'un pressupost realista

3. **Redisseny** (**19/09/2014-05/10/2014**)
    a. Solucionar problema de incompatibilitat entre Arduino i Windows 7 64-bits
    b. Finalització del codi de la llibreria implementada
    c. Finalització del codi .ino

4. **Muntatge** (**06/10/2014-08/10/2014**)
    a. Muntatge de la part electrònica de l'instrument
    b. Testeig del funcionament
    c. Plantejament de possibles millores en funció del temps disponible

5. **Disseny de la carcassa de l'instrument i compra de materials** (**09/10/2014-11/10/2014**)
    a. Disseny de la carcassa i planificació de la distribució dels components
    b. Cerca i compra de la carcassa i els materials necessaris per a la seva manipulació

6. **Muntatge** (**12/10/2014**)
    a. Manipulació de la carcassa en funció del disseny establert
    b. Inserció dels components electrònics
    c. Rectificació de mesures i repetició del procés de manipulació en el cas de cometre algun error

7. **Redacció de la memòria** (**13/10/2014-20/10/2014**)

Objectius assolits i conclusions

S'ha seguit el pla de treball de manera rigorosa, i s'ha redactat de manera més realista, en base al temps disposat i la feina disponible ja anteriorment realitzada. Aquests fets han implicat un compliment bastant precís dels diferents punts, dins dels deadlines proposats.

Per contra, cal tenir en compte que per tal d'assolir els temps decidits, s'ha hagut de prescindir del desenvolupament de prestacions que es contemplaven inicialment en el pla de treball, i s'ha vist obligat a traspassar a l'apartat "Futures línies de treball".

Futures línies de treball

Ampliació de la llibreria RTP

RTPCalibrationSeq

Una possible eina molt útil dins de la llibreria, seria implementar diversos mètodes i classes per tal què qualsevol instrument dissenyat i programat utilitzant la llibreria RTP, pogués calibrar els seus sensors per així regular els possibles rangs que poguessin abastar. Un exemple clar on es podria aplicar, seria en l'encesa de l'unoStringSynth, on al iniciar el setup(), invoca la funció CalibrationSeq(), on captura el valor màxim i mínim produït per la inducció de la pastilla del baix elèctric, per a poder mapejar correctament els límits amb els rangs de la placa Arduino. L'objectiu d'aquesta eina, seria aconseguir que pogués ser genèrica per a qualsevol tipus de sensor, de manera que només es treballés amb senyals d'entrada i sortida.

RTPLedControl

L'unoStringSynth disposa d'un codi de colors, que indica la duració del temps de calibratge de l'aparell i l'estat del seu procés, al igual que el nombre d'octaves en què el màstil es subdivideix i, per tant, que l'instrument pot arribar a reproduir. Si es volgués treballar amb leds per tal d'estipular diversos codis de colors que informessin de diversos estats o valors, una bona eina a desenvolupar seria un codi genèric per tal d'aplicar-lo als instruments dissenyats.

RTPSwitchControl

Al igual que l'ús de leds indiquen missatges o estats, també serà habitual introduir valors o canviar paràmetres a través de switchos booleans a fi de voler configurar l'instrument dissenyat d'una manera adient en funció de les seves prestacions. En cas que el nombre de polsadors s'incrementés considerablement, seria estudiable desenvolupar una eina que agrupés tots els switchos i funcions disponibles, i fins i tot programar algun tipus de combinatòria de pulsació dels botons, a fi de poder accedir a altres paràmetres de configuració, i així reduir costos de fabricació estalviant botons, sense renunciar a totes les possibles prestacions que es volguessin oferir dins de l'instrument dissenyat.

Desenvolupament de prestacions echoTheremin

Seleccionador de nombre d'octaves

Afegint un polsador més a la carcassa i implementant dins del codi les modificacions pertinents, una eina molt útil que trencaria limitacions sonores, seria un switch que decidís la quantitat d'octaves per les que es divideix tot el recorregut que és capaç de detectar el sensor de posició encarregat de la selecció de notes. Només seria necessari treballar amb múltiples sobre l'array dels salts de les escales i sobre els steps que les conformen, per així multiplicar el nombre de notes possibles a executar en un mateix recorregut i subdividir l'espai que ocupen pel nombre total de notes factibles.

Enviament de missatges MIDI

Tot i estar implementada la funció dins de la llibreria RTPMidiSet, en aquest prototip no s'ha arribat a instaurar conjuntament amb el codi .ino de l'echoTheremin. Una bona manera de augmentar exponencialment les possibilitats d'aquest instrument, seria finalitzar el procés per a configurar les senyals que genera l'arduino convertides en senyals midi, a fi de poder fer generar les mateixes notes que executem amb la mà apropant o allunyant-la del sensor de posició, a altres màquines com poden ser sintetitzadors o ordinadors.

Implementació de control de guany

Les sortides digitals de les plaques arduino, únicament generen senyals de 0 i 5V, sense possibilitat de graduar aquesta sortida de voltatge. Però disposa de diverses sortides PWM[19], capaces de oferir resultats analògics a través de senyals digitals.

Aquestes sortides permeten la manipulació del temps d'activació i desactivació del flanc de la senyal quadrada que emeten a una freqüència constant, i que en aquest cas, coincideix amb la de la nota que l'instrument genera. Dit d'un altre manera, permeten la manipulació temporal de la part positiva de la senyal, sense manipular la freqüència d'oscil·lació.

[19] Pulse-width modulation

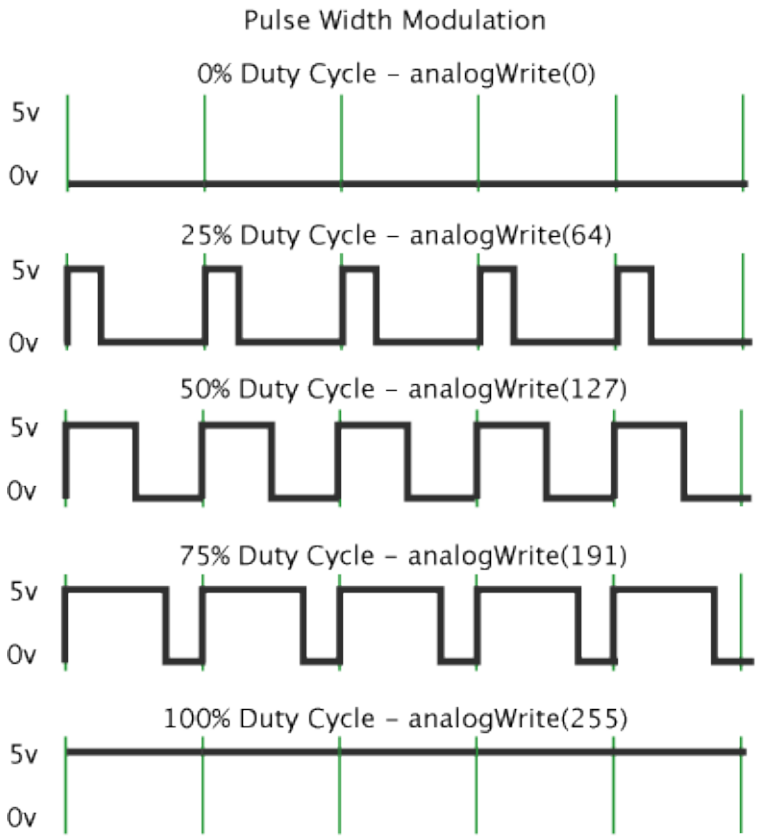


fig. 22: exemple de funcionament del PWM

Treballar amb aquest tipus de sortides dotaria al echoTheremin de valor interpretatiu, al poder manipular la intensitat de les notes generades amb la mateixa precisió i la mateixa metodologia que com ara mateix es seleccionen les notes musicals (mitjançant el sensor de proximitat per ultrasons lateral).

Millora en les possibilitats sonores

El treball i l'estudi més intensiu de les sortides PWM, ofereix la possibilitat de generar ones sinusoïdals a diferents amplituds i freqüències, sumar-ne per a crear harmònics, i fins i tot crear efectes com poden ser delays, tremolos, flangers...

Una opció a tenir en compte per a millorar aquest aspecte de l'instrument, seria treballar amb altres llibreries ja implementades, per a combinar la execució i metodologia de l'instrument, amb les possibilitats sonores ja implementades, com per exemple, la llibreria Mozzi.

A la seva pàgina web (inclosa dins de la bibliografia d'aquest projecte) s'hi poden trobar diversos exemples dels sons, filtres i efectes que es poden arribar a generar, tutorials de diversos nivells, i fins i tot la possibilitat de contribuir en el projecte mitjançant l'accés al seu GitHub[20].

[20] Sistema virtual de control col·laboratiu de revisió i desenvolupament de software

Disseny i carcassa

Dins de la presentació del projecte, seria interessant millorar el disseny de la carcassa de l'instrument, per tal de millorar la seva resistència davant possibles cops o sotracs, mantenint el compromís d'obtenir la màxima lleugeresa possible.

També seria útil plantejar alternatives de suport a la adaptació duta a terme en aquest prototip, per tal de tantejar la possibilitat d'oferir altres suports que es poguessin acoblar en altres instruments, com podrien ser sintetitzadors o instruments de percussió.

Pedal de sustain

Un dels inconvenients d'aquest prototip, pel seu disseny, és que la seva execució demana l'ús de les dues mans, incompatibilitzant la variació dels valors d'algun paràmetre i la execució de l'instrument simultània.

fig. 23: pedal de sustain de teclat elèctric

Com a solució o alternativa, es podria dissenyar un pedal de sustain, simulant els pedals de piano, per a que es pogués mantenir activada la última nota executada, per tal de poder apartar la mà del sensor d'intensitat, i així disposar de la mà dreta lliure per tal de poder variar els paràmetres. En cas de desenvolupar aquesta extensió, seria convenient tenir en compte que la variació dels paràmetres no s'executés fins deixar de prémer el botó de sustain.

Variació d'afinació

Segons les necessitats de l'intèrpret, sovint es vol modificar l'afinació del propi instrument (ja sigui per a ser fidel a la interpretació d'obres antigues, on la afinació podia estar estipulada a un valor diferent als 440Hz corresponents al A4 actual, o bé per a experimentar). És per a aquest motiu, que es treballa només amb coeficients que multipliquen una freqüència base, com és en el cas del codi echoTheremin.ino, els 440Hz.

Mitjançant un selector més per tal de poder incrementar o disminuir aquest valor, mitjançant un get-set es podria manipular aquest valor per a ser precís a l'hora de configurar l'afinació exacte de l'instrument.

Conclusions

Impacte social, musical i tecnològic

La creació d'un instrument nou com és l'echoTheremin, aporta una nova perspectiva tant dins del món musical com tecnològic. En aquest cas, uns senzills càlculs ja estipulats dins d'una llibreria que serveix com a eina per a crear instruments musicals, aconsegueixen que una persona que no ha estudiat mai música, o que mai ha tocat un instrument musical, pugui interactuar i crear peces, conjuntament amb altres músics, sense patir el procés poc agraït d'aprendre a tocar un instrument des de 0.

En termes reals, inicialment aquest instrument tot i ser d'ús intuïtiu, cal una certa tècnica per a poder repetir dues melodies exactament iguals. Així, com tot instrument, necessita una fase d'instrucció i aprenentatge, amb la diferència que al ser tan intuïtiu, la corba d'aprenentatge és molt més agraïda que aprendre a tocar altres instruments musicals com bé poden ser la guitarra o el violí.

Un altre ús que es pot aplicar a l'echoTheremin, en aquest cas, és més didàctic i lúdic. Comercialitzar-lo com a joguina educativa musical, aportaria una nova via d'apropament als estudis musicals, a nens i nenes petits, ja que en edats tempranes està comprovada l'atracció i facilitat d'aprenentatge, i l'execució de sons i escales ja corregides i per tant correctes, molt probablement aportaria millores i es guanyaria temps d'assaig/error per part dels infants quan aprenen a desenvolupar el seu oïda musical.

Finalment, i possiblement la utilitat més realista d'aquest instrument, és la possibilitat al músic professional de l'ús d'una nova eina de creació, on les lleis físiques de la manipulació d'un instrument, i les limitacions tècniques de cadascun, queden resoltes a partir de la tecnologia Arduino. Tant a nivell interpretatiu, com d'execució, pot esdevenir una eina clau per a la generació de noves idees musicals treballant a partir de la aleatorietat i la intuïció, a més de la oportunitat d'explotar diferents sons i possibilitats no convencionals.

Conclusions finals

La realització d'aquest projecte a aportat nous coneixements de diversos camps de la programació, la tecnologia Arduino, i el moviment maker a l'hora d'aconseguir relacionar els estudis i nocions adquirides durant l'enginyeria cursada, amb els principals interessos que, en aquest cas, són purament musicals.

La creació de la llibreria RTPLibrary no és més que una petita pinzellada de les possibilitats que ofereix aquesta tecnologia, i l'oportunitat de compartir aquests coneixements i objectius assolits, a altres persones que vulguin millorar i continuar dins d'aquest projecte, per a aconseguir millors resultats i generar nous instruments amb més prestacions i més possibilitats.

La funcionalitat d'aquesta llibreria, que és oferir la possibilitat a qualsevol persona de crear un instrument musical, únicament plantejant-se com haurà de ser executat per a la seva interpretació, és factible. I la creació de l'instrument echoTheremin, n'és el testimoni.

fig. 24: resultat final de l'echoTheremin

Bibliografia

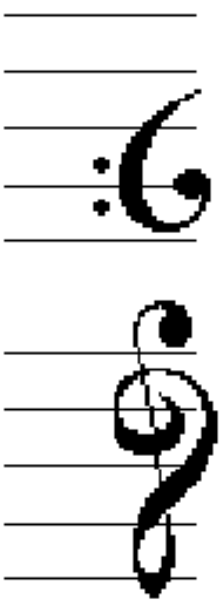
- *10 Ways to Destroy An Arduino*, Michigan (USA) (document en línia http://www.ruggedcircuits.com/10-ways-to-destroy-an-arduino/ - consultada el 19/10/14)
- Arduino, pàgina web oficial, 2014 (web en línia http://www.arduino.cc/ - consultada el 19/10/14)
- *Arduino*, wikipedia (web en línia http://en.wikipedia.org/wiki/Arduino - consultada el 19/10/14)
- Daniel Gallardo García (Profesor de Tecnología del IES Laguna de Tollón), *Apuntes de Arduino "Nivel Pardillo"* (document en línia http://educacionadistancia.juntadeandalucia.es/profesorado/pluginfile.php/2882/mod_resource/content/1/Apuntes_ARDUINO_nivel_PARDILLO.pdf - consultat el 19/10/14)
- *Diatonic Scale*, wikipedia (web en línia http://en.wikipedia.org/wiki/Diatonic_scale - consultada el 19/10/14)
- Diotronic, pàgina web oficial (web en línia http://www.diotronic.com/ - consultada el 19/10/14)
- *Escala menor armónica*, wikipedia (web en línia http://es.wikipedia.org/wiki/Escala_menor_arm%C3%B3nica - consultada el 19/10/14)
- *Escala musical*, wikipedia (web en línia http://es.wikipedia.org/wiki/Escala_musical - consultada el 19/10/14)
- *Escala pentatònica*, wikipeda (web en línia http://es.wikipedia.org/wiki/Escala_pentat%C3%B3nica - consultada el 19/10/14)
- *Escalas exóticas para improvisar,* El blog de la educación musical (web en líniahttp://elblogdelaeducacionmusical.blogspot.com.es/2010/06/escalas-exoticas-para-improvisar.html - consultada el 19/10/14)
- *Las Escalas Exóticas*, Hispasonic (web en línai http://www.hispasonic.com/foros/escalas-exoticas/368212 - consultada el 19/10/14)
- *Maker Culture*, wikipedia (web en línia http://en.wikipedia.org/wiki/Maker_culture - consultada el 19/10/14)
- Martínez, Óscar. *unoStringSynth,* Blog del projecte (web en línia http://unostringsynth.blogspot.com.es/ - consultada el 19/10/14)
- Mozzi, pàgina web ofical (web en línia http://sensorium.github.io/Mozzi/ - consultada el 19/10/14)
- *Musical Instrument Digital Interface (MIDI),* wikipeda (web en líniahttp://en.wikipedia.org/wiki/MIDI - consultada el 19/10/14)
- *Note names, MIDI numbers and frequencies,* Music Acoustics, School of Physics at UNSW, Sydney (web en línia http://newt.phys.unsw.edu.au/jw/notes.html - consultada el 19/10/14)
- *Phone connector (audio),* wikipedia (web en linia http://en.wikipedia.org/wiki/Phone_connector_%28audio%29 - consultada el 19/10/14)

- Protocol de missatges MIDI (web en línia http://www.midi.org/techspecs/midimessages.php - consultada el 19/10/14)
- *Pulsadores para microcontrolador usando entradas digitales* (web en línia http://txapuzas.blogspot.com.es/2010/07/pulsadores-para-microcontrolador.html - consultada el 19/10/14)
- *Pulse Width Modulation*, All About Circuits, 2014 (web en línia http://www.allaboutcircuits.com/vol_3/chpt_11/1.html - consultada el 19/10/14)
- *Pulse-width modulation,* FabCentral, Motion System Design, n.101, October 2000 (document en línia http://fab.cba.mit.edu/classes/MIT/961.04/topics/pwm.pdf - consultada el 19/10/14)
- *Pulse-width modulation,* wikipeda (web en línia http://en.wikipedia.org/wiki/Pulse-width_modulation - consultada el 19/10/14)
- Rockin' Tech Projects , pàgina web oficial (web en línia http://www.rockintechprojects.net/ - consultada el 19/10/14)
- Scale Arrays, Eigenlabs, 2010 (web en línia http://www.eigenlabs.com/wiki/2.0/Scale_Arrays/ - consultada el 19/10/14)
- Sparkfun Shop (web en línia https://www.sparkfun.com/ - consultada el 19/10/14)
- *SRF05 Ultra-Sonic Ranger, Technical Specification*, Robo electronics (web en línia http://www.robot-electronics.co.uk/htm/srf05tech.htm - consultada el 19/10/14)
- Theremin Word Website (web en línia http://www.thereminworld.com/ - consultada el 19/10/14)
- *Tutorial Arduino #003 – Entrada Analógica y Salida PWM,* Arduteka (web eb línia http://www.arduteka.com/2011/11/tutorial-arduino-0003-entrada-analogica-y-salida-pwm/ - consultada el 19/10/14)
- *Ultrasonic Distance Sensor Product Guide*, Parallax – Ping))) (document en línia http://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf - consultada el 19/10/14)
- *Ultrasonic Ranging Module HC-SR04 Datasheet,* Elecfreaks (document en línia http://users.ece.utexas.edu/~valvano/Datasheets/HCSR04b.pdf - consultada el 19/10/14)

Annexos

1. Taula de relació notes musicals

MIDI number	Note name	Keyboard	Frequency Hz	Period ms
21	A0		27.500	36.36
22	A#0 / Bb0		29.135	34.32
23	B0		30.868	32.40
24	C1		32.703	30.58
25	C#1 / Db1		34.648	28.86
26	D1		36.708	27.24
27	D#1 / Eb1		38.891	25.71
28	E1		41.203	24.27
29	F1		43.654	22.91
30	F#1 / Gb1		46.249	21.62
31	G1		48.999	20.41
32	G#1 / Ab1		51.913	19.26
33	A1		55.000	18.18
34	A#1 / Bb1		58.270	17.16
35	B1		61.735	16.20
36	C2		65.406	15.29
37	C#2 / Db2		69.296	14.29
38	D2		73.416	13.62
39	D#2 / Eb2		77.782	12.86
40	E2		82.407	12.13
41	F2		87.307	11.45
42	F#2 / Gb2		92.499	10.81
43	G2		97.999	10.20
44	G#2 / Ab2		103.83	9.631
45	A2		110.00	9.091
46	A#2 / Bb2		116.54	8.581
47	B2		123.47	8.099
48	C3		130.81	7.645
49	C#3 / Db3		138.59	7.216
50	D3		146.83	6.811
51	D#3 / Eb3		155.56	6.428
52	E3		164.81	6.068
53	F3		174.61	5.727
54	F#3 / Gb3		185.00	5.405
55	G3		196.00	5.102
56	G#3 / Ab3		207.65	4.816
57	A3		220.00	4.545
58	A#3 / Bb3		233.08	4.290
59	B3		246.94	4.050
60	**C4**		**261.63**	**3.822**
61	C#4 / Db4		277.18	3.608
62	D4		293.67	3.405
63	D#4 / Eb4		311.13	3.214
64	E4		329.63	3.034
65	F4		349.23	2.863
66	F#4 / Gb4		369.99	2.703
67	G4		392.00	2.551
68	G#4 / Ab4		415.30	2.408
69	**A4**		**440.00**	**2.273**
70	A#4 / Bb4		466.16	2.145
71	B4		493.88	2.025
72	C5		523.25	1.910
73	C#5 / Db5		554.37	1.804
74	D5		587.33	1.703
75	D#5 / Eb5		622.25	1.607
76	E5		659.26	1.517
77	F5		698.46	1.432
78	F#5 / Gb5		739.99	1.351
79	G5		783.99	1.276
80	G#5 / Ab5		830.61	1.204
81	A5		880.00	1.136
82	A#5 / Bb5		932.33	1.073
83	B5		987.77	1.012
84	C6		1046.5	0.9556
85	C#6 / Db6		1108.7	0.9020
86	D6		1174.7	0.8513
87	D#6 / Eb6		1244.5	0.8034
88	E6		1318.5	0.7584
89	F6		1396.9	0.7159
90	F#6 / Gb6		1480.0	0.6757
91	G6		1568.0	0.6378
92	G#6 / Ab6		1661.2	0.6020
93	A6		1760.0	0.5682
94	A#6 / Bb6		1864.7	0.5362
95	B6		1975.5	0.5062
96	C7		2093.0	0.4778
97	C#7 / Db7		2217.5	0.4510
98	D7		2349.3	0.4257
99	D#7 / Eb7		2489.0	0.4018
100	E7		2637.0	0.3792
101	F7		2793.0	0.3580
102	F#7 / Gb7		2960.0	0.3378
103	G7		3136.0	0.3189
104	G#7 / Ab7		3322.4	0.3010
105	A7		3520.0	0.2841
106	A#7 / Bb7		3729.3	0.2681
107	B7		3951.1	0.2531
108	C8	J. Wolfe, UNSW	4186.0	0.2389



2. Codi Arduino unoStringSynth Original

```
/*
unoStringSynth: Sketch per a programar el firmware d'Arduino com a "standalone".
Ideat i Desenvolupat per: Óscar Martínez Carmona (Nasker) - 2012/2013
 */

const int octDownButton = 9;
const int octUpButton = 8;

const int octLed1 =  10;
const int octLed2 =  11;
const int octLed3 =  12;
const int octLed4 =  13;

const int envPWM = 6;
const int outTone = 7;

int envMin = 1023;
int envMax = 0;
int maxVal=1023;

float bendReadMapped = 0;
int bendReadCenter = 0;

//boolean growing = false;

//float envMapPast = 0;

float envTreshold;
boolean state = false;
boolean prevState = state;
float prevPitch=0;

int octDownState = 0;
int octUpState = 0;

int nOct = 4;
int oStep = 0;
int midiChanel = 0x90;

int octStep[] = {1, 2, 4, 8, 16, 32, 64};

int octaveSwitch = 0;

float diatonicStep[12][7] ={
  {1,1.1225,1.2599,1.3348,1.4983,1.6818,1.8877}, //1-Ionian
  {1,1.1225,1.1892,1.3348,1.4983,1.6818,1.7818}, //2-Dorian
  {1,1.0595,1.1892,1.3348,1.4983,1.5874,1.7818}, //3-Phrygian
  {1,1.1225,1.2599,1.4142,1.4983,1.6818,1.8877}, //4-Lydian
  {1,1.1225,1.2599,1.3348,1.4983,1.6818,1.7818}, //5-Mixolydian
  {1,1.1225,1.1892,1.3348,1.4983,1.5874,1.7818}, //6-Aeolian
  {1,1.0595,1.1892,1.3348,1.4142,1.5874,1.7818}, //7-Locrian
  {1,1.1225,1.1892,1.3348,1.4983,1.5874,1.8877}, //8-Harmonic Minor
  {1,1.0595,1.2599,1.3348,1.4983,1.5874,1.7818}, //9-Spanish Gipsy
  {1,1.1225,1.1892,1.3348,1.4983,1.6818,1.8877}, //10-Hawaian
```

```
  {1,1.1892,1.3348,1.4142,1.4983,1.7818}, //11-Blues
  {1,1.0595,1.3348,1.4983,1.5874} //12-Japanese
};

int nSteps;


// the setup routine runs once when you press reset:
/*
const int bufSize = 1 ;
float buffer[bufSize];
int bufInd = 0;
int bufMeanPast = 0;
*/

void setup() {

  Serial.begin(31250);
  //Serial.begin(9600);

  pinMode(octLed1, OUTPUT);
  pinMode(octLed2, OUTPUT);
  pinMode(octLed3, OUTPUT);
  pinMode(octLed4, OUTPUT);

  pinMode(envPWM, OUTPUT);

  pinMode(outTone, OUTPUT);

  pinMode(octUpButton, INPUT);
  pinMode(octDownButton, INPUT);

  calibrationSeq();
}


// the loop routine runs over and over again, and again, and again, and again...
void loop() {

  int tonalitySwitch = round ( float(analogRead(A2)) /1023 * 11);

    if (tonalitySwitch < 10)  nSteps = 7;
    else if (tonalitySwitch == 10) nSteps = 6;
    else nSteps = 5;

  int rootSwitch = round ( float(analogRead(A1)) /1023 * 11);

  float rootFreq = 440 * pow(2,(-45+rootSwitch)/12.);

  int envMapped = map(analogRead(A3), envMin, envMax, 0, 255);

  envMapped = constrain(envMapped, 0, 255);

  octControl();

  ledControl();
```

```
  int nScaleSteps = nSteps;

  float AnNormRib = float(analogRead(A0)) /1023;

  int octRibbon = floor(AnNormRib * float(nOct));

  int stepRibbon = ceil(AnNormRib * (nOct * nScaleSteps))-1;

  if (stepRibbon <= 0) stepRibbon = 0;

  int stepScale = stepRibbon - (nScaleSteps * octRibbon);

  int octCorr = 0;
  if (octRibbon >= 2 ) octCorr = 1;

  int octPowd = pow(2,octRibbon)+octCorr;

  //Calcul de la freq del to generat en funcio de la fonamental, la lectura
  //del sweetPot, l'escala i l'octava. <<PEDRA ANGULAR D'AQUEST PROJECTE!>>
  float pitch = rootFreq * diatonicStep[tonalitySwitch][stepScale] * octPowd
*octStep[oStep];

  float bendRead = analogRead(A4);

  if(bendRead<20) bendReadMapped=-1;
  if((bendRead>=20)&&(bendRead<=460)) bendReadMapped = (1-(bendRead-20)/440)*-1;
  if((bendRead>460)&&(bendRead<580)) bendReadMapped=0;
  if((bendRead>=580)&&(bendRead<=1005)) bendReadMapped = (bendRead-580)/425;
  if(bendRead>1005)bendReadMapped=1;


 int pitchBendRange=12;

  float pitchBend=pow(2,pitchBendRange*bendReadMapped/12);

  tone(outTone, pitch*pitchBend);

  int midPitchBend = round((pitchBend+1)*64-1);

  if(pitchBend!=0) midiNoteOn(224,0,midPitchBend);

  analogWrite(envPWM, envMapped);

  //Serial.println(pitchBend);

  if(envMapped > 40) state=true;
  else {
    state=false;
    midiNoteOn(midiChanel,pitch,0x00);
  }

  int velocity = int(envMapped/2);

  if(state!=prevState){
    if(state){
      //Serial.println("NOTE_ON");
      //delay(5);
```

```
      int velocity = int(envMapped/2);
      midiNoteOn(midiChanel,pitch,velocity);
    }
    if(!state){
      //Serial.println("NOTE_OFF");
      midiNoteOn(midiChanel,pitch,0x00);
    }
    prevState=state;
    if(prevPitch!=pitch){
      midiNoteOn(midiChanel,prevPitch,0x00);
      midiNoteOn(midiChanel,pitch,velocity);
    }
    prevPitch=pitch;
  }

}
  /*
  buffer[bufInd] = envMapped;
  float bufMean=0;
  bufInd++;

  if(bufInd>=(bufSize-1)){
    bufMean = bufMeanCalc(buffer);
    bufInd=0;
    //Serial.println(bufMean);
  }



 if (bufMean > envMax/10){
    if(bufMean >= bufMeanPast) growing = true;
    else if ( growing && bufMean < bufMeanPast){
      growing = false;
      int velocity = round(map(bufMeanPast,0, 255, 0, 127));
      midiNoteOn(midiChanel,pitch,velocity);
      delay(20);
    }
    bufMeanPast = bufMean;
  }
  else{
  midiNoteOn(midiChanel,pitch,0x00);
  delay(20);
  }
}


/*

 if (envMapped > envTreshold){
    if(envMapped >= envMapPast) growing = true;
    else if ( growing && envMapped < envMapPast*1.1){
      growing = false;
      int velocity = round(map(envMapPast,envMin, envMax, 0, 127));
      midiNoteOn(0X90,pitch,velocity);
      delay(20);
    }
    envMapPast = envMapped;
```

```
  }
  midiNoteOn(0x90,pitch,0x00);
  delay(20);
 }
}

float bufMeanCalc(float buffer[]){
  float mean=0;
  for(int i=0; i<bufSize; i++) mean+=buffer[i];
  return mean/bufSize;
}
*/
void midiNoteOn(int channel, int pitch, int velocity){

  int midiNote = round(12*log2(pitch/440.))+69;

  Serial.write(channel);
  Serial.write(midiNote);
  Serial.write(velocity);
}


float log2 (float x) {
  return (log(x) / log(2));
}


void calibrationSeq(){

  bendReadCenter=(maxVal+1)/2;
  while (millis() < 8000) {
    int sensorValue = analogRead(A3);

    if (sensorValue > envMax) {
      envMax = sensorValue;
    }

    if (sensorValue < envMin) {
      envMin = sensorValue;
    }
    if (millis() > 0001) digitalWrite(octLed1,HIGH);
    if (millis() > 1000) digitalWrite(octLed2, HIGH);
    if (millis() > 2000) digitalWrite(octLed3, HIGH);
    if (millis() > 3000) digitalWrite(octLed4, HIGH);
    if (millis() > 4000) digitalWrite(octLed1, LOW);
    if (millis() > 5000) digitalWrite(octLed2, LOW);
    if (millis() > 6000) digitalWrite(octLed3, LOW);
    if (millis() > 7000) digitalWrite(octLed4, LOW);
  }

  envTreshold = envMax/4;

  //envTreshold = map(envTreshold, envMin, envMax, 0, 255);

  //Serial.println(envTreshold);
}
```

```
void octControl(){

  octUpState = digitalRead(octUpButton);
  octDownState = digitalRead(octDownButton);

  if (octUpState == HIGH) {
    octaveSwitch++;
    delay(50);
  }
  if (octDownState == HIGH ) {
    octaveSwitch--;
    delay(50);
  }

  if (octaveSwitch > 6) octaveSwitch = 6;
  if (octaveSwitch < 0) octaveSwitch = 0;
}


void ledControl (){

   switch (octaveSwitch) {
  case 0:
    digitalWrite (octLed1, HIGH);
    digitalWrite (octLed2, LOW);
    digitalWrite (octLed3, LOW);
    digitalWrite (octLed4, LOW);
    oStep = 0;
    nOct = 4;
    midiChanel = 0x90;
    break;
  case 1:
    digitalWrite (octLed1, HIGH);
    digitalWrite (octLed2, HIGH);
    digitalWrite (octLed3, LOW);
    digitalWrite (octLed4, LOW);
    oStep = 1;
    nOct = 4;
    midiChanel = 0x90;
    break;
  case 2:
    digitalWrite (octLed1, LOW);
    digitalWrite (octLed2, HIGH);
    digitalWrite (octLed3, LOW);
    digitalWrite (octLed4, LOW);
    oStep = 2;
    nOct = 4;
    midiChanel = 0x90;
    break;
  case 3:
    digitalWrite (octLed1, HIGH);
    digitalWrite (octLed2, HIGH);
    digitalWrite (octLed3, HIGH);
    digitalWrite (octLed4, HIGH);
    oStep = 0;
    nOct = 8;
    midiChanel = 0x90;
```

```
    break;
  case 4:
    digitalWrite (octLed1, LOW);
    digitalWrite (octLed2, LOW);
    digitalWrite (octLed3, HIGH);
    digitalWrite (octLed4, LOW);
    oStep = 0;
    nOct = 4;
    midiChanel = 0x91;
    break;
  case 5:
    digitalWrite (octLed1, LOW);
    digitalWrite (octLed2, LOW);
    digitalWrite (octLed3, HIGH);
    digitalWrite (octLed4, HIGH);
    oStep = 1;
    nOct = 4;
    midiChanel = 0x91;
    break;
  case 6:
    digitalWrite (octLed1, LOW);
    digitalWrite (octLed2, LOW);
    digitalWrite (octLed3, LOW);
    digitalWrite (octLed4, HIGH);
    oStep = 2;
    nOct = 4;
    midiChanel = 0x91;
    break;
  }

}
```

3. Codi Arduino unoStringSynth revisat

```
/*
unoStringSynth: Sketch per a programar el firmware d'Arduino com a "standalone".
Ideat i Desenvolupat per: Óscar Martínez Carmona (Nasker) - 2012/2013
Revisat per: Óscar Martínez Carmona (Nasker) i Oriol Corroto Sales - 2013/2014
 */

//S'asignen els pins digitals 8 i 9 per als polsadors de canvi d'octava
const int switchDownControl = 9;  //nºpin --octava
const int switchUpControl = 8;    //nºpin ++octava

//s'asignaran als pins digitals 10, 11, 12 i 13 (LEDS)
const int octLed1 =  10;
const int octLed2 =  11;
const int octLed3 =  12;
const int octLed4 =  13;

const int envelopePWM = 6; // Assignem el pin Digital 6 per a la PWM de l'envolvent
const int toneOut = 7; //Assignem el pin Digital 7 per a la sortida del to generat

const int mastil = 0;
const int crTonalitat = 1;
const int crEscala = 2;
const int envelopePIN = 3;
const int pitchBendIn= 4;
const int pitchReference = 440;

#define analMax 1023  //Valor analògic màxim
int envMin = 1023;    //al valor mínim registrat se li otorga un rang de 1023 (VALOR
INICIAL SETUP)
int envMax = 0;       //al valor màxim registrat se li otorga un rang de 0 (VALOR
INICIAL SETUP)

//inicialització de valors del pitch-bend
float bendReadMapped = 0;
int bendReadCenter = 0;

//VARIABLES MIDI
float envelopeTreshold;
boolean midiState = false;
boolean preMidiState = midiState;
float prevPitch=0;
int midiChannel = 0x90;

boolean switchDownState = false;  //L'estat inicial dels polsadors per canviar
d'octava es defineix com a nul
boolean switchUpState = false;

int octaveNumber = 4;  //subdivisions en octaves del màstil per defecte
int octaveStep = 0;  //exponent de 2^n per a seleccionar octava

int controlSwitch = 0;  //oS es el index per a recorrer les caselles de l'array
d'octaves, s'inicia al centre del rang (8)
```

```
float diatonicStep[12][12] ={
  {1,1.0595,1.1225,1.1892,1.2599,1.3348,1.4142,1.4983,1.5875,1.6818,1.7818,1.8877},
//1-Chromatic
  {1,1.1225,1.2599,1.3348,1.4983,1.6818,1.8877}, //2-Ionian
  {1,1.1225,1.1892,1.3348,1.4983,1.6818,1.7818}, //3-Dorian
  {1,1.0595,1.1892,1.3348,1.4983,1.5874,1.7818}, //4-Phrygian
  {1,1.1225,1.2599,1.4142,1.4983,1.6818,1.8877}, //5-Lydian
  {1,1.1225,1.2599,1.3348,1.4983,1.6818,1.7818}, //6-Mixolydian
  {1,1.1225,1.1892,1.3348,1.4983,1.5874,1.7818}, //7-Aeolian
  {1,1.0595,1.1892,1.3348,1.4142,1.5874,1.7818}, //8-Locrian
  {1,1.1225,1.1892,1.3348,1.4983,1.5874,1.8877}, //9-Harmonic Minor
  {1,1.0595,1.2599,1.3348,1.4983,1.5874,1.7818}, //10-Spanish Gipsy
  {1,1.1892,1.3348,1.4142,1.4983,1.7818}, //11-Blues
  {1,1.0595,1.3348,1.4983,1.5874} //12-Japanese
};
//Array d'arrays el qual conte la distribucio de factors de multiplicacio per obtenir
els graus (en 2 octaves) dins
//de cadascuna de les 12 escales

int nSteps; //Nombre de graus dins de l'escala, emprat per quantitzar el valor
continu de la cinta del mastil

void setup() {

  //clase de comunicació serie
  //Serial.begin(31250);      //31250 -> velocitat de transmissio del MIDI
  Serial.begin(9600);     //9600 -> velocitat de transmissio del Monitor Serial

  pinMode(octLed1, OUTPUT);  //Es configuren els pins digitals per als LED's com a
sortida
  pinMode(octLed2, OUTPUT);
  pinMode(octLed3, OUTPUT);
  pinMode(octLed4, OUTPUT);

  pinMode(envelopePWM, OUTPUT);

  pinMode(toneOut, OUTPUT);  //El pins digitals  5, 6 i 7 s'empren per a la sortida
dels tons

  pinMode(switchUpControl, INPUT);  //Els pins asignats als posaldors son configurats
com a entrades
  pinMode(switchDownControl, INPUT);

  calibrationSeq();

}


// the loop routine runs over and over again, and again, and again, and again...
void loop() {

//COMMUTADOR ROTATIU ESCALA
  int tonalitySwitch = map(analogRead(crEscala),0, analMax,0,11);
    if (tonalitySwitch == 0) nSteps=12;
//fraccions de notes en funció de la escala
    else if (tonalitySwitch < 10 && tonalitySwitch > 0)  nSteps = 7;
    else if (tonalitySwitch == 10) nSteps = 6;
```

```
    else nSteps = 5;

//COMMUTAdOR ROTATIU TONALITAT
  int rootSwitch = map(analogRead(crTonalitat),0, analMax,0,11);   //0 = C , 11 = B

  float rootFreq = pitchReference * pow(2,(-45+rootSwitch)/12.);  //-36 = 3
octavesDown -9 = 9 salts restants == -45

  int envMapped = map(analogRead(envelopePIN), envMin, envMax, 0, 255);

  envMapped = constrain(envMapped, 0, 255);  //constrenyiment dels valors d'envolvent

  switchControl(); //funció de control d'octaves (switchos)

  ledControl();  //funcio que preestableix els PRESETS

  int nScaleSteps = nSteps;

  float AnNormRib = float(analogRead(mastil)) /1023;  //valor obtingut del SoftPot
Membrane normalitzat

  int octRibbon = floor(AnNormRib * float(octaveNumber));  //retorn de la octava
corresponent

  int stepRibbon = ceil(AnNormRib * (octaveNumber * nScaleSteps))-1;  // aconseguir
el valor exacte de la nota (alça(posicioRibbon*(totaloctaves*saltsescala))-1)
                                                                    //quantificador
del recorregut del Ribbbon
  if (stepRibbon <= 0) stepRibbon = 0;  //si dona valor negatiu, igualem a 0

  int stepScale = stepRibbon - (nScaleSteps * octRibbon); //retorna el valor on s'ha
d'apuntar de la matriu a partir de la posició del Ribbon

  int octPowd = 1 << octRibbon;

  int octaveMultiplier = 1 << octaveStep;

  //Calcul de la freq del to generat en funcio de la fonamental, la lectura
  //del softPot, l'escala i l'octava. <<PEDRA ANGULAR D'AQUEST PROJECTE!>>
  float pitch = rootFreq * diatonicStep[tonalitySwitch][stepScale] * octPowd
*octaveMultiplier;



 //PITCH (floyd rose :P)
  float bendRead = analogRead(pitchBendIn);

  if(bendRead<20) bendReadMapped=-1;
  if((bendRead>=20)&&(bendRead<=460)) bendReadMapped = (1-(bendRead-20)/440)*-1;
  if((bendRead>460)&&(bendRead<580)) bendReadMapped=0;
  if((bendRead>=580)&&(bendRead<=1005)) bendReadMapped = (bendRead-580)/425;
  if(bendRead>1005)bendReadMapped=1;

  int pitchBendRange=12;

  float pitchBend=pow(2,pitchBendRange*bendReadMapped/12);
```

```
  tone(toneOut, pitch*pitchBend);

  analogWrite(envelopePWM, envMapped);

  //PITCH PER MIDI (només funcina decreixentment)
  int midPitchBend = round((pitchBend+1)*64-1);

  if(pitchBend!=0) midiNoteOn(224,0,midPitchBend);


  if(envMapped > 40) midiState=true;
  else {
    midiState=false;
    midiNoteOn(midiChannel,pitch,0x00);
  }

  int velocity = int(envMapped/2);

  if(midiState!=preMidiState){
    if(midiState){
      midiNoteOn(midiChannel,pitch,velocity);
    }
    if(!midiState){
      midiNoteOn(midiChannel,pitch,0x00);
    }
    preMidiState=midiState;
    if(prevPitch!=pitch){
      midiNoteOn(midiChannel,prevPitch,0x00);
      midiNoteOn(midiChannel,pitch,velocity);
      prevPitch=pitch;
    }
  }
}

void midiNoteOn(int channel, int pitch, int velocity){

  int midiNote = round(12*log2(pitch/440.))+69;

  Serial.write(channel);
  Serial.write(midiNote);
  Serial.write(velocity);
}


float log2 (float x) {
  return (log(x) / log(2));
}


void calibrationSeq(){

  bendReadCenter=(analMax+1)/2;
  while (millis() < 8000) {
    int sensorValue = analogRead(A3);   //entrada analog A3

    if (sensorValue > envMax) {         //calibració de la envolvent (valor màxim)
      envMax = sensorValue;
```

```
    }

    if (sensorValue < envMin) {         // calibració de la envolvent (valor mínim)
      envMin = sensorValue;
    }
    if (millis() > 0001) digitalWrite(octLed1,HIGH);
    if (millis() > 1000) digitalWrite(octLed2, HIGH);
    if (millis() > 2000) digitalWrite(octLed3, HIGH);
    if (millis() > 3000) digitalWrite(octLed4, HIGH);
    if (millis() > 4000) digitalWrite(octLed1, LOW);
    if (millis() > 5000) digitalWrite(octLed2, LOW);
    if (millis() > 6000) digitalWrite(octLed3, LOW);
    if (millis() > 7000) digitalWrite(octLed4, LOW);
  }

  envelopeTreshold = envMax/4;
}

void switchControl(){  //funció de control d'octaves (switchos)

  switchUpState = digitalRead(switchUpControl);
  switchDownState = digitalRead(switchDownControl);

  if (switchUpState == HIGH) {
    controlSwitch++;
    delay(100);
  }
  if (switchDownState == HIGH ) {
    controlSwitch--;
    delay(100);
  }

  if (controlSwitch > 6) controlSwitch = 6; //limitació de rang
  if (controlSwitch < 0) controlSwitch = 0; //limitacio de rang
}


void ledControl (){  //interficie "gràfica" per controlar els modes

   switch (controlSwitch) {
  case 0:  //PRESET 1
    digitalWrite (octLed1, HIGH);
    digitalWrite (octLed2, LOW);
    digitalWrite (octLed3, LOW);
    digitalWrite (octLed4, LOW);
    octaveStep = 0;
    octaveNumber = 4;
    midiChannel = 0x90;  //NoteOn Midi Message
    break;
  case 1:  //PRESET 2
    digitalWrite (octLed1, HIGH);
    digitalWrite (octLed2, HIGH);
    digitalWrite (octLed3, LOW);
    digitalWrite (octLed4, LOW);
    octaveStep = 1;
    octaveNumber = 4;
    midiChannel = 0x90;  //NoteOn Midi Message
```

```
    break;
  case 2:  //PRESET 3
    digitalWrite (octLed1, LOW);
    digitalWrite (octLed2, HIGH);
    digitalWrite (octLed3, LOW);
    digitalWrite (octLed4, LOW);
    octaveStep = 2;
    octaveNumber = 4;
    midiChannel = 0x90;  //NoteOn Midi Message
    break;
  case 3:  //PRESET 4
    digitalWrite (octLed1, HIGH);
    digitalWrite (octLed2, HIGH);
    digitalWrite (octLed3, HIGH);
    digitalWrite (octLed4, HIGH);
    octaveStep = 0;
    octaveNumber = 8;
    midiChannel = 0x90;  //NoteOn Midi Message
    break;
  case 4:  //PRESET 5
    digitalWrite (octLed1, LOW);
    digitalWrite (octLed2, LOW);
    digitalWrite (octLed3, HIGH);
    digitalWrite (octLed4, LOW);
    octaveStep = 0;
    octaveNumber = 4;
    midiChannel = 0x91;
    break;
  case 5:  //PRESET 6
    digitalWrite (octLed1, LOW);
    digitalWrite (octLed2, LOW);
    digitalWrite (octLed3, HIGH);
    digitalWrite (octLed4, HIGH);
    octaveStep = 1;
    octaveNumber = 4;
    midiChannel = 0x91;
    break;
  case 6:  //PRESET 7
    digitalWrite (octLed1, LOW);
    digitalWrite (octLed2, LOW);
    digitalWrite (octLed3, LOW);
    digitalWrite (octLed4, HIGH);
    octaveStep = 2;
    octaveNumber = 4;
    midiChannel = 0x91;
    break;
  }

}
```

4. Codi RTPDiatonicMatrix.h

```
#ifndef RTPDiatonicMatrix_h
#define RTPDiatonicMatrix_h

#include "Arduino.h"

class RTPDiatonicMatrix{


  int _toneStep[13][13];
  String _scaleName[13];
// Array de coeficients de semitons equivalents a cada escala.
  int _tonality;
  int _stepScale;
  int _nSteps;
  float _coef;
  int _coeficient;
  int _getcheck;
  int _numberScales;

       public:                          //A public hi declarem els mètodes que volem
emprar desde fora de la classe!
              RTPDiatonicMatrix();         //Mètode constructor(imprescindible per
inicialitzar la classe!
    void setTonality(int tonality);
    int getSteps();
    float getCoef(int stepScale);
    int getCheck();
    int getNumberScales();
    String getScaleName(int scale);

            //Resta de mètode(s) public(s)

  private:

};

#endif
```

5. Codi DiatonicMatrix.cpp

```
#include "Arduino.h"
#include "RTPDiatonicMatrix.h"


RTPDiatonicMatrix::RTPDiatonicMatrix(){
int toneStep[13][13] = {
    {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}, //0-Chromatic
    {0, 2, 4, 5, 7, 9, 11, 12}, //1-Ionian
    {0, 2, 3, 5, 7, 9, 10, 12}, //2-Dorian
    {0, 1, 3, 5, 7, 8, 10, 12}, //3-Phrygian
    {0, 2, 4, 6, 7, 9, 11, 12}, //4-Lydian
    {0, 2, 4, 5, 7, 9, 10, 12}, //5-Mixolydian
    {0, 2, 3, 5, 7, 8, 10, 12}, //6-Aeolian
    {0, 1, 3, 5, 6, 8, 10, 12}, //7-Locrian
    {0, 2, 3, 5, 7, 8, 11, 12}, //8-Harmonic Minor
    {0, 1, 4, 5, 7, 8, 10, 12}, //9-Spanish Gipsy
    {0, 2, 3, 5, 7, 9, 11, 12}, //10-Hawaian
    {0, 3, 5, 6, 7, 10, 12}, //11-Blues
    {0, 1, 5, 7, 8, 12} //12-Japanese
       };

String scaleName[13] ={
    "Chromatic", "Ionian", "Dorian", "Phrygian",
    "Lydian","Mixolydian", "Aeolian", "Locrian",
    "Harmonic", "Gipsy", "Hawaian", "Blues", "Japanese"};

for (int i=0; i<13; i++){
    for (int j=0; j<13; j++){
        _toneStep[i][j] = toneStep[i][j];
    }
    _scaleName[i]=scaleName[i];
}
// Array de coeficients de semitons equivalents a cada escala.
_tonality = 0;
_stepScale = 0;
_coef = 0;
_nSteps = 0;
_coeficient = 0;
_numberScales = 13;
}

void RTPDiatonicMatrix::setTonality(int tonality){
    _tonality = tonality;
}

int RTPDiatonicMatrix::getSteps(){
if (_tonality == 0) _nSteps=12;        //fraccions de notes en funció de la escala
    else if (_tonality < 10 && _tonality > 0)  _nSteps = 7;
    else if (_tonality == 10) _nSteps = 6;
    else _nSteps = 5;
    return _nSteps;
}

float RTPDiatonicMatrix::getCoef(int stepScale){
```

```
_stepScale = stepScale;
_coeficient= _toneStep[_tonality][_stepScale];
_getcheck= _coef;
return _coeficient;
}

String RTPDiatonicMatrix::getScaleName(int scale){
    return _scaleName[scale];
}

int RTPDiatonicMatrix::getNumberScales(){
    return _numberScales;
}
```

6. Codi RTPSelectTone.h

```
#ifndef RTPSelectTone_h
#define RTPSelectTone_h

#include "Arduino.h"

class RTPSelectTone{


  int _toneFreq[12];
  String _toneName[12];
// Array de valors freqüencials d'una octava.
  int _note;
  int _stepnote;
  int _nSteps;
  int _freq;


       public:                          //A public hi declarem els mètodes que volem
emprar desde fora de la classe!
               RTPSelectTone();         //Mètode constructor
    void setFreq(int note);
    int getFreq(int stepnote);
    String getToneName(int tone);

             //Resta de mètode(s) public(s)

  private:

};

#endif
```

7. Codi RTPSelectTone.cpp

```
#include "Arduino.h"
#include "RTPSelectTone.h"


RTPSelectTone::RTPSelectTone(){
int toneFreq[12] = {
    0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };

String toneName[12] ={
    "A", "A#", "B", "C",
    "C#","D", "D#", "E",
    "F", "F#", "G", "G#"};

for (int i=0; i<12; i++){
    _toneFreq[i] = toneFreq[i];
    _toneName[i] = toneName[i];
}
// Array de salts cromàtics equivalents a les opcions de tonalitat.
_note = 0;
_stepnote = 0;
_freq = 0;
_nSteps = 0;
}

void RTPSelectTone::setFreq(int note){
    _note = note;
}


int RTPSelectTone::getFreq(int stepnote){
    _stepnote = stepnote;
    _freq= _toneFreq[_note];
    return _freq;
}

String RTPSelectTone::getToneName(int tone){
    return _toneName[tone];
}
```

8. Codi RTPPitchControl.h

```
#ifndef RTPPitchControl_h
#define RTPPitchControl_h

#include "Arduino.h"

class RTPPitchControl{

int _pitchBendRange;
int _bendReadCenter;
float _bendRead;
float _bendReadMapped;
float _pitchBendOut;

       public:                          //A public hi declarem els mètodes que volem
emprar desde fora de la classe!
               RTPPitchControl();         //Mètode constructor
               float getPitchBend(float bendRead);
             //Resta de mètode(s) public(s)
               void setPitchRange(int rangeSemitones);

  private:

};

#endif
```

9. Codi RTPPitchControl.cpp

```
#include "Arduino.h"
#include "RTPPitchControl.h"

RTPPitchControl::RTPPitchControl(){
       _pitchBendRange = 12;
       _bendReadCenter = 0;
    _bendRead = 0.0;
    _bendReadMapped = 0.0;
    _pitchBendOut = 0.0;
}

float RTPPitchControl::getPitchBend(float bendRead){
       _bendRead = bendRead;
       if(_bendRead<20) _bendReadMapped=-1;
    else if((_bendRead>=20)&&(_bendRead<=460)) _bendReadMapped = (1-(_bendRead-
20)/440)*-1;
    else if((_bendRead>460)&&(_bendRead<580)) _bendReadMapped=0;
    else if((_bendRead>=580)&&(_bendRead<=1005)) _bendReadMapped = (_bendRead-
580)/425;
    else _bendReadMapped=1;

    _pitchBendOut = pow(2,_pitchBendRange*_bendReadMapped/12);
    return _pitchBendOut;
}

void setPitchRange(int rangeSemitones){
       _pitchBendRange = rangeSemitones;
}
```

10. Codi RTPMidiSet.h

```
#ifndef RTPMidiSet_h
#define RTPMidiSet_h

#include "Arduino.h"
#include "RTPMaths.h"

class RTPMidiSet{
       int _channel;
       int _note;
       int _velocity;
       float _startFreq;
       int _startNote;


       public:                          //A public hi declarem els mètodes que volem
emprar desde fora de la classe!
               RTPMidiSet();        //Mètode constructor
               float getChannel();
               float getNote();
               float getVelocity();
               void setChannel(int channel);
               void setNote(int startNote);
               void setVelocity(int velocity);
               void midiNoteOn(int channel, int note, int velocity);
              //Resta de mètode(s) public(s)

  private:

};

#endif
```

11. Codi RTPMidiSet.cpp

```
#include "Arduino.h"
#include "RTPMidiSet.h"
#include "RTPMaths.h"

RTPMidiSet::RTPMidiSet(){
	_channel= 0x90;
	_note= 0;
	_velocity=0x45;
	_startFreq= 440.;
	_startNote=69;
	RTPMaths instancia1;
}

void RTPMidiSet::setChannel(int channel){
	_channel=channel;
}

void RTPMidiSet::setNote(int startNote){
	_startNote=startNote;
}

void RTPMidiSet::setVelocity(int velocity){
	_velocity=velocity;
}

float RTPMidiSet::getChannel(){
	return _channel;
}

float RTPMidiSet::getNote(){
	return _startNote;
}

float RTPMidiSet::getVelocity(){
	return _velocity;
}

void RTPMidiSet::midiNoteOn(int channel, int note, int velocity){
	_note= round(12*instancia1.log2(note/_startFreq))+_startNote;
	_channel=channel;
	_velocity=velocity;
	Serial.write(_channel);
	Serial.write(_note);
	Serial.write(_velocity);
}
```

12. Codi RTPMaths.h

```
#ifndef RTPMaths_h
#define RTPMaths_h

#include "Arduino.h"

class RTPMaths{


       public:                          //A public hi declarem els mètodes que volem
emprar desde fora de la classe!
              RTPMaths();         //Mètode constructor
       float log2(float x);

             //Resta de mètode(s) public(s)

  private:

};

#endif
```

13. Codi RTPMaths.cpp

```
#include "Arduino.h"
#include "RTPMaths.h"

RTPMaths::RTPMaths(){
}

float RTPMaths::log2(float x){
       return (log(x) / log(2));
}
```

14. Codi echoTheremin.ino

```
#include <Wire.h> //llibreria destinada a la comunicació i2C
#include <LiquidCrystal_I2C.h> //llibreria destinada a la impresso del display
#include <Ultrasonic.h> //llibreria destinada als sensors d'ultrafreqüència
#include <NewTone.h> //llibreria per a poder generar tons
#include <RTPDiatonicMatrix.h>
#include <RTPSelectTone.h>

LiquidCrystal_I2C lcd(0x27,16,2); //instància display
RTPDiatonicMatrix instancia1; //instància RTPDiatonicMatrix
RTPSelectTone instancia2; //instància RTPSelectTone


const int speakerOut = 7;  //valor de sortida de so
const int pushIn = 8;      //valor de entrada de boto d'activació
const int pushScale = 6;   //valor d'entrada de boto de canvi de escala
const int pushTone = 5;    //valor d'entrada de boto de canvi de tó
const int midiOut = 9;     //valor de sortida de les dades MIDI
String name;              //string que conté el nom de la escala seleccionada
String nametone;          //string que conté el nom del tó seleccionat

Ultrasonic ultrasonic1(1,0); // (Trig PIN,Echo PIN)
Ultrasonic ultrasonic2(4,3); // (Trig PIN,Echo PIN)
unsigned int noteUltrasonicRead;
unsigned int gainUltrasonicRead;

/*
//valors inicialitzats pels BUFFERS de nota i gain
int mitja1; int mitja2;
int c1=0; int c2=0;
int buffer1[5]; int buffer2[5];
*/
//delcaració de nota i intensitat
float noteFreq; float noteGain;

int scale =0; //iniciació del valor escala (Cromàtica)
int fnote =0; //iniciació del valor tonalitat (A4)
float afinacio = 440; //paràmetre de calibratje d'afinació


void setup() {
//clase de comunicació serie
    Serial.begin(9600); //velocitat de transmissió del Monitor Serial
    pinMode(speakerOut,OUTPUT); //pin de sortida d'audio
    pinMode(pushIn,INPUT); //pin d'entrada d'acció de nota
    pinMode(pushScale,INPUT); //pin d'entrada de canvi d'escala
    pinMode(pushTone, INPUT); //pin d'entrada de canvi de tonalitat

  /* Initialise the LCD */
  lcd.init();
  lcd.init();
  lcd.clear();
  /* Make sure the backlight is turned on */
  lcd.backlight();
  /* Output the test message to the LCD */
```

```
  lcd.setCursor(0,0); //fila 0, columna 0
  lcd.print("***SCALE TYPE***");
  lcd.setCursor(0,1); //fila 1, columna 0
  lcd.print("Chromatic");
}


void loop() {
  /*manipulació del paràmetre per a canviar d'escala*/
        instancia1.setTonality(scale);
        boolean Escala = digitalRead(pushScale);
        if (Escala){
          delay(100); scale++; //delay per a canvi de valor
          name= instancia1.getScaleName(scale);
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("***SCALE TYPE***");
          lcd.setCursor(0,1); lcd.print(name);
        }
        if (scale>12) { //inici al primer valor de l'array
          scale=0;
          name= instancia1.getScaleName(scale);
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("***SCALE TYPE***");
          lcd.setCursor(0,1);
          lcd.print(name);
        }
  /*manipulació del paràmetre per a canviar de to*/
         instancia2.setFreq(fnote);
         boolean Frequencia = digitalRead(pushTone);
         if (Frequencia){
             delay(100); fnote++; //delay per a canvi de valor
             nametone= instancia2.getToneName(fnote);
             lcd.clear();
             lcd.setCursor(0,0);
             lcd.print("***TONE TYPE***");
             lcd.setCursor(0,1); lcd.print(nametone);
         }
         if (fnote>11){ //inici al primer valor de l'array
             fnote=0;
             nametone= instancia2.getToneName(fnote);
             lcd.clear();
             lcd.setCursor(0,0);
             lcd.print("***TONE TYPE***");
             lcd.setCursor(0,1); lcd.print(nametone);
         }

  noteUltrasonicRead = ultrasonic1.Ranging(CM); //obtenció del valor del sensor de
posició de la nota
  //gainUltrasonicRead = ultrasonic2.Ranging(CM); //obtenció del valor del sensor de
posició d'intensitat


/*  //BUFFER PER A PROMITJAR ELS VALORS DEL SENSOR DE POSICIÓ
     buffer1[c] = noteUltrasonicRead;
     c++;
```

```
    if (c == 5) {
      c = 0;
      mitja = 0;
      for (int i = 0; i < 5; i++) mitja = mitja + buffer[i];
      mitja = mitja / 5;
      // introduir codi executor
    }
*/
//CODI EXECUTOR
       int dStep = map(noteUltrasonicRead,0,51,0,instancia1.getSteps());
       float coef=pow(2,float (instancia1.getCoef(dStep)/12.));
       noteFreq = afinacio*(pow(2,float (instancia2.getFreq(fnote)/12.))) * coef;
 //
    delay(25); //delay de correcció

       boolean button = digitalRead(pushIn);
       if(button)
         NewTone(speakerOut, noteFreq);
       else
         noNewTone(speakerOut);
}
```